(19)日本国特許庁（ＪＰ）　(12)公表特許公報（Ａ）　(11)特許出願公表番号
特表2002－525755
（P2002－525755A）
(43)公表日　平成14年８月13日(2002.8.13)

(51)Int.Cl.[7]	識別記号	FI	テーマコード(参考)
G06F 12/00	513	G06F 12/00	513A 5B075
	501		501B 5B082
	535		535Z
17/30	180	17/30	180D
	240		240A

審査請求　未請求　予備審査請求　有　（全 60 頁）

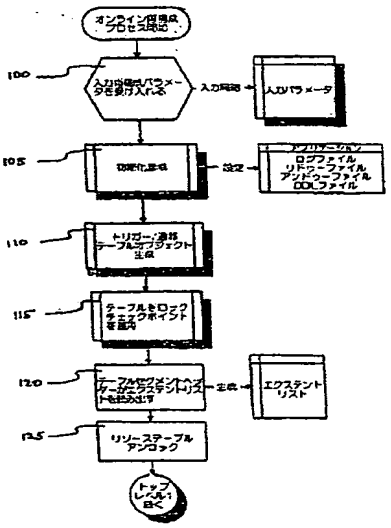
(21)出願番号　特願2000－571377(P2000－571377)
(86)(22)出願日　平成11年９月22日(1999.9.22)
(85)翻訳文提出日　平成13年３月22日(2001.3.22)
(86)国際出願番号　ＰＣＴ／ＵＳ９９／２２０４４
(87)国際公開番号　ＷＯ００／１７７８７
(87)国際公開日　平成12年３月30日(2000.3.30)
(31)優先権主張番号　０９／１５９，０７３
(32)優先日　平成10年９月22日(1998.9.22)
(33)優先権主張国　米国（ＵＳ）
(81)指定国　ＥＰ(ＡＴ，ＢＥ，ＣＨ，ＣＹ，ＤＥ，ＤＫ，ＥＳ，ＦＩ，ＦＲ，ＧＢ，ＧＲ，ＩＥ，ＩＴ，ＬＵ，ＭＣ，ＮＬ，ＰＴ，ＳＥ），ＪＰ

(71)出願人　コンピュータ　アソシエイツ　シンク，インコーポレイテッド
アメリカ合衆国・ニューヨーク州
11749・イスランディア・ワン　コンピュータ　アソシエイツ　プラザ
(72)発明者　ペレイラ　ヒルトン　エム
アメリカ合衆国　カリフォルニア州
91360　サウザンド　オークス　サフロン　サークル　3092
(74)代理人　弁理士　中村　稔　（外９名）

Ｆターム(参考)　5B075 NK02 NR02 QT06
5B082 CA11 FA18 GA07

(54)【発明の名称】　アクティブＤＢＭＳテーブルを再編成するための方法及び装置

(57)【要約】
データベース表の再編成が行われ、再編成された表が正常のＯＬＴＰアクティビティに利用可能である。トリガーが設定され、再編成されたソース表（１１０）のＯＬＴＰアクティビティを記憶する。ソース表は一瞬ロックされ、ＳＣＮチェックポイントを確率すると共にソース表（１１５）のために割当てられたデータブロックの位置を決定する。タイムスタンプはソース表からアンロードされると共に新しい表（１２０）に挿入されるので、表のコピーは（新しい表）及び触れられていない／変更されたブロックを作り出される。タイムスタンプが新しい表で行われてからトランザクションが起こり、タイムスタンプがトリガーレコードの状態となってからトランザクションがもう起こらないようになっている。新しい表が（ソース表として改名された）ソース表により切換えられ、元のソース表はドロップされる。



【特許請求の範囲】

【請求項１】　ソーステーブルのオンライン認識方法であって、

ポイントインタイム（時間内のポイント）と該ポイントインタイムでの前記ソーステーブルのステートを識別するタイムスタンプを確立するステップと、

前記タイムスタンプでの前記ソーステーブルの再編成コピーを生成するステップと、

前記タイムスタンプ後に生じるトランザクションを前記再編成テーブルに適用するステップと、

前記ソーステーブルを前記再編成テーブルにスイッチするステップと

から成ることを特徴とする方法。

【請求項２】　請求項１に記載の方法において、更に、

前記ソーステーブルのトランザクションのエントリをトランザクションテーブルにログするために、前記ソーステーブルのトリガを利用するステップと、

前記ソーステーブルにトランザクションが生じるのを阻止するために前記ソーステーブルにロックを獲得するステップと、

前記ソーステーブルの前記ロックを獲得する前に前記トリガによってログされたエントリを削除するステップと、

前記ソーステーブルのロックを解除するステップとを備え、

前記トリガの利用ステップと前記ロックの獲得ステップとは、前記タイムスタンプの確立ステップよりも前に行われる

ことを特徴とする方法。

【請求項３】　請求項２に記載の方法において、前記ソーステーブルの再編成コピーを生成するステップが、

前記再編成コピーを維持するために新オブジェクトを生成するステップと、

前記タイムスタンプ以後には修正されない前記ソーステーブルのブロックからデータをアンロードするステップと、

前記データを前記新オブジェクトにロードするステップと

を包含することを特徴とする方法。

【請求項４】　請求項３に記載の方法において、前記トランザクションを適

用するステップが、

前記ソーステーブルに排他的ロックを獲得するステップと、

以前にアンロードされた、前記ソーステーブルのブロックに対応する、前記トランザクションテーブルからのトランザクションを、前記新オブジェクトにロードされた前記データに適用するステップと、

前記タイムスタンプ以後のトランザクションに含まれる前記ソーステーブルのブロックであって以前にはアンロードされていないブロックからデータをアンロードし、そのデータを前記新オブジェクトにロードするステップと、

前記ソーステーブルの前記排他的ロックを解放するステップと

から成ることを特徴とする方法。

【請求項５】　請求項３に記載の方法において、前記トランザクションを適用するステップが、

前記ソーステーブルに排他的ロックを獲得するステップと、

前記新オブジェクトに以前にロードされ且つ前記トランザクションテーブルにおいて識別されたブロックに対応するデータを削除するステップと、

前記ソーステーブルの修正データを、前記トランザクションテーブルにおいて以前にアンロードされて識別されたブロックからアンロードし、前記修正データを前記新オブジェクトにロードするステップと、

前記タイムスタンプ以後のトランザクションに含まれる前記ソーステーブルのブロックであって以前にはアンロードされていないブロックからデータをアンロードし、そのデータを前記新オブジェクトにロードするステップと、

前記ソーステーブルの前記排他的ロックを解放するステップと

から成ることを特徴とする方法。

【請求項６】　請求項２に記載の方法において、前記トランザクションを適用するステップが、

前記ソーステーブルに排他的ロックを獲得するステップと、

前記トランザクションログの残りのエントリを、前記再編成テーブルに適用するステップと、

前記排他的ロックの結果として、いずれかの追加のトランザクションがペンデ

ィング中であるかどうかを判定し、追加のトランザクションがペンディングである場合、

前記ソーステーブルの前記排他的ロックを解放するステップと、

追加のトランザクションがペンディングでなくなるまで、前記排他的ロックの獲得ステップと前記残りのエントリの適用ステップと前記判定ステップとを繰り返すステップとを行うステップと

から成ることを特徴とする方法。

【請求項７】　請求項６に記載の方法において、前記タイムスタンプの確立ステップが、

前記タイムスタンプとして使用するためのシステム変更番号（System Change Number）を決定するステップを含み、該システム変更番号は、前記ソーステーブル及び該ソーステーブルに関連して維持されたテーブル又はオブジェクトのいずれか１つに生じるトランザクションについてのユニークな識別子を提供する

ことを特徴とする方法。

【請求項８】　請求項１に記載の方法において、前記ソーステーブルはデータベーステーブルであることを特徴とする方法。

【請求項９】　請求項８に記載の方法において、前記再編成コピーを生成するステップが、

前記再編成コピーを維持するため新データベーステーブルオブジェクトを生成するステップと、

前記タイムスタンプ以後にタッチされていない前記ソースデータベーステーブルからのテーブル情報のブロックを読取るステップと、

前記読取ったブロックから行情報を抽出するステップと、

前記行情報を前記新データベーステーブルオブジェクトにストアするステップと

から成ることを特徴とする方法。

【請求項１０】　前記新しいデータベーステーブルオブジェクトを作るステップは、

ソースデータベーステーブルのために、データベースデータ定義言語（ＤＤＬ

）スキームを検索し、

前記ＤＤＬを利用して、前記新しいデータベーステーブルオブジェクトを作るステップを含み、

前記再編成したコピーを作るステップは、参照位置への索引付けシステムを確立するステップを含み、前記行情報は、前記ソースデータベーステーブルから検索され、前記新しいデータベーステーブルオブジェクトに記憶される請求項９に記載の方法。

【請求項１１】　コンピュータ命令を記憶していて、コンピュータにロードされるとき、コンピュータに、

時間のポイントからなるタイムスタンプを設立し、前記時間のポイントにおける前記ソーステーブルの状態を識別し、

前記ソーステーブルの再編成されたコピーを作り、

前記タイムスタンプ後に起こるトランザクションを前記再編成されたテーブルに適用し、

前記ソーステーブルを前記再編成されたテーブルと共に切換えるステップを実行させることを特徴とするコンピュータ可読媒体。

【請求項１２】　前記記憶された命令は、コンピュータに、

トランザクションテーブル内で前記ソーステーブル上のトランザクションのエントリをログするため、前記ソーステーブル上のトリガーを用い、

前記ソーステーブル上のロックを得て、前記ソーステーブルにトランザクションが起こるのを防止し、

前記ソーステーブル上の前記ロックを得る前に、前記トリガーによりログされたエントリを削除し、

前記ソーステーブルをアンロックするステップを実行させ、

前記トリガーを用いまたロックを得るステップは、前記タイムスタンプを確立するステップより前に行われる請求項１１に記載のコンピュータ可読媒体。

【請求項１３】　前記ソーステーブルの再編成されたコピーを作る前記ステップは、

前記再編成されたコピーを保持するため、新しいオブジェクトを作り、

前記ソーステーブルのブロックから、前記タイムスタンプから修正されていないデータをアンロードし、

前記データを前記新しいオブジェクトにロードするステップを含む請求項１２に記載のコンピュータ可読媒体。

【請求項１４】　前記トランザクションを適用するステップは、

前記ソーステーブル上に排他的ロックを得て、

以前にアンロードされた前記ソーステーブルのブロックに対応する前記トランザクションテーブルから、前記新しいオブジェクトにロードされた前記データに、トランザクションを適用し、

前記チェックポイントからトランザクションに含まれているが以前にはアンロードされていないデータを、前記ソーステーブルのブロックからアンロードし、それを前記新しいオブジェクトにロードし、

前記ソーステーブル上の排他的ロックを解放するステップを含む請求項１３に記載のコンピュータ可読媒体。

【請求項１５】　前記トランザクションを適用するステップは、

前記ソーステーブル上に排他的ロックを得て、

以前に前記新しいオブジェクトにロードされ、前記トランザクションテーブル内に識別されたブロックに対応するデータを削除し、

以前にアンロードされ前記トランザクションテーブル内に識別されたブロックから、前記ソーステーブルの修正されたデータをアンロードし、前記修正されたデータを前記新しいオブジェクトにロードし、

前記タイムスタンプからトランザクションに含まれているが以前にはアンロードされていないデータを、前記ソーステーブルのブロックからアンロードし、それを前記新しいオブジェクトにロードし、

前記ソーステーブル上の排他的ロックを解放するステップを含む請求項１３に記載のコンピュータ可読媒体。

【請求項１６】　前記トランザクションを適用するステップは、

前記ソーステーブル上に排他的ロックを得て、

前記トランザクションログの残りのエントリを前記認識されたテーブルに適用

し、

前記排他的ロックの結果として、任意の別のトランザクションがペンディングになっているか求め、もし別のトランザクションがペンディングになっていれば、

前記ソーステーブル上の排他的ロックを解放し、

前記排他的ロックを得て、残りのエントリを適用し、求めるステップを繰り返し、ペンディングになっている別のトランザクションがなくなるまで行うステップを含む請求項１２に記載のコンピュータ可読媒体。

【請求項１７】　前記タイムスタンプを確立するステップは、前記ソーステーブルおよび他のテーブルまたは前記ソーステーブルに関連して維持されるオブジェクトのうちの任意の１つに対して起こるトランザクションに対する独特な識別子を与えるシステム変更番号を、前記タイムスタンプとして使用するのに決定するステップを含む請求項１６に記載のコンピュータ読み取り可能な媒体。

【請求項１８】　前記ソーステーブルは、データベーステーブルであり、前記再組織コピーを生成するステップは、前記再組織コピーを維持するため新しいデータベースオブジェクトを生成するステップと、前記タイムスタンプ以後触れられていないソースデータベーステーブルからのテーブル情報のブロックを読み取るステップと、前記読み取られたブロックから行情報を抽出するステップと、前記行情報を前記新しいデータベーステーブルオブジェクトに記憶させるステップとを含む請求項１に記載のコンピュータ読み取り可能な媒体。

【請求項１９】　前記新しいテーブルオブジェクトを生成するステップは、前記データベーステーブルのためのデータベースデータ定義言語（ＤＤＬ）スキーマを検索するステップと、前記ＤＤＬを使用して前記新しいテーブルオブジェクトを生成するステップとを含み、前記再組織コピーを生成するステップは、更に、前記行情報が前記データベーステーブルから検索され前記新しいテーブルオブジェクトに記憶される基準場所に対するインデキシングシステムを確立するステップを含む請求項１８に記載のコンピュータ読み取り可能な媒体。

【請求項２０】　前記Ｓ.ＣＮチェックポイントを確立するステップは、トランザクションテーブルにおける前記ソーステーブルに関するトランザクションの

エントリーをロギングするため前記ソーステーブルに関しトリガを使用し、トランザクションが前記ソーステーブルに対して起きないようにするため前記ソーステーブルに関しロックを得て、前記ソーステーブルに関し前記ロックを得る前に、前記トリガによってロギングされたエントリーを削除し、前記ソーステーブルをアンロックすることを含み、前記トリガを使用しロックを得るステップは、前記タイムスタンプを確立するステップの前に行われる請求項１８に記載のコンピュータ読み取り可能な媒体。

【請求項２１】　ソースファイルのオンライン再組織装置において、ある時間点からなり該時間点での前記ソースファイルの状態を識別するＳＣＮチェックポイントを確立するように構成されたチェックポイントデバイスと、前記ソースファイルの再組織コピーを生成するように構成されたコピー機構とを備えており、前記タイムスタンプの後に起こるトランザクションを前記再組織ファイルに適用し、前記ソースファイルを前記再組織ファイルと交換することを特徴とするソースファイルのオンライン再組織装置。

【請求項２２】　更に、トランザクションテーブルにおける前記ソースファイルに関するトランザクションのエントリーをロギングするように構成されたトリガーと、トランザクションが前記ソースファイルに対して起きないようにするため前記ソースファイルに関しロックを得るように構成されたロッキング機構と、前記ソースファイルに関し前記ロックを得る前に前記トリガーによってロギングされたエントリーを削除するように構成された除去デバイスと、前記ソースファイルをアンロックするように構成されたアンロッキング機構とを備える請求項２１に記載の装置。

【請求項２３】　前記コピー機構は、前記再組織コピーを維持するため新しいオブジェクトを生成するように構成された生成デバイスと、前記タイムスタンプ以後変更されていない前記ソースファイルのブロックからデータをアンロードするように構成されたアンローディング機構と、前記データを前記新しいオブジェクトへロードするように構成されたローディング機構とを備える請求項２２に記載の装置。

【請求項２４】　前記トランザクションデバイスは、前記新しいオブジェク

トへロードされたデータへ適用されるべく、前にアンロードされた前記ソースファイルのブロックに対応するトランザクションを選択するように構成された選択機構と、前記タイムスタンプ以後トランザクションに含まれていて前にアンロードされていない前記ソースファイルのブロックからのデータをアンロードし該データを前記新しいオブジェクトへロードするように構成されたアンローディング／ローディングデバイスとを備える請求項２３に記載の装置。

【請求項２５】前記の処理装置は、削除機構とアンロード/ロード装置とを備え、

前記の削除機構は、前記の新しい対象の中に以前に入れられ、そして前記の処理表内で特定されたブロックに対応するデータを削除するように構成されており、そして

前記のアンロード/ロード装置は、以前にアンロードされ、そして前記の処理表内で特定されたブロックから前記のソース･ファイルの変更されたデータをアンロードし、そしてその変更したデータを前記の新しい対象の中へロードするよう構成され、そして前記のタイムスタンプ以来処理に含まれ、そして以前にアンロードされていない前記のソース･ファイルのブロックからデータをアンロードし、そしてそれを前記の新しい対象の中へロードするよう構成されている
請求項２３に記載の装置。

【請求項２６】　前記の処理装置は、

前記の処理ログ内の残っているエントリーを再編成したファイルへ加えるよう構成されたアプリケータ；

前記のソース表がロックされた結果として追加の処理が継続中かどうかを決めるよう構成された判定機構；

ソースファイル上の前記のイクスクルーシブ･ロックを繰り返しレリーズし、前記のソースファイル上にイクスクルーシブ･ロックを得、そして前記のロッキング機構、アンロッキング機構、アプリケータそして判定機構を介して付加的な処理が継続していることがなくなるまで前記の処理ログに残りのエントリーを加えるよう構成されたコントローラ
を備えている請求項２２に記載の装置。

【請求項２７】　前記のタイムスタンプは、前記のソースファイルの一つに、そして前記のソースファイルと関連した他のファイルもしくは対象に生じている処理を特定するアイデンティファイアを与えるシステム・チェンジ・ナンバーである請求項２６に記載の装置。

【請求項２８】　前記のソースファイルはデータベース表であり、そして前記のコピー機構は、

前記の再編成されたコピーを保持するための新しい表である対象をつくるよう構成された対象装置；

前記のデータベースファイルから表情報のブロックを読むように構成された読取機構；

読まれたブロックから行情報を抽出するよう構成された抽出装置；そして

新しい表対象に前記の行情報を蓄えるよう構成された蓄積装置

を備えた請求項２１に記載の装置。

【請求項２９】　前記の対象装置は、前記のデーターベース表のためのデーターベースデーター定義語スキーマ（ＤＤＬ）を検索し、そしてこのＤＤＬを使って前記の新しい表対象をつくるよう構成されたＤＤＬ反復子を備え、

前記の装置は更に、前記の行情報が前記のデーターベース表から検索され、そして前記の新しい表対象に蓄えられている場所を参照するよう構成されたインデックサを備えている請求項２８に記載の装置。

【請求項３０】　前記のチェックポイント装置は更に、

前記のデーターベース表上の処理のエントリーを処理表にログするよう構成されたトリガー；

処理が前記のソース表に生じないように前記のデーターベース表上でロックするよう構成されたロッキング機構；

前記のデーターベース表上でロックする前に前記のトリガーによりログされたエントリーを削除するよう構成された除去装置；そして

前記のデーターベース表をアンロックするよう構成されたアンロッキング機構

を備えた請求項２９に記載の装置。

【請求項３１】　ポイント・イン・タイムを備えるタイムスタンプを確立し

、そして前記のポイント･イン･タイムにおけるソース表の状態を特定する手段；

前記のタイムスタンプの直前の状態において前記のソース表の再編成されたコピーをつくる手段；

前記のチェックポイントの後生じる処理を再編成された表へ加える手段；そして

前記のソース表を再編成された表と切り替える手段

を備えたソース表のオン・ライン再編成装置。

【請求項３２】　前記のソース表上にトリガーを使用し、ソース表上の処理のエントリーを処理表にログ･インする手段；

前記のソース表上にロックを得て処理が前記のソース表へ生じないようにする手段；

前記のソース表上に前記のロックを得る前に前記のトリガーによりログされたエントリーを削除すす手段；そして

前記のソース表をアンロックする手段

を更に備える請求項３１に記載の装置。

【請求項３３】　前記の再編成された表を前記のソース表と切り替えた後前記のソース表を落とす手段を更に備えている請求項３１に記載の装置。

【請求項３４】　前記のソース表がデーターベース表であり、そして前記の再編成されたコピーをつくる手段が、

前記の再編成されたコピーを保持するため新しい表対象をつくる手段、

前記のデーターベース表から表情報のブロックを読み取る手段；

読み取られたブロックから行情報を抽出する手段；そして

新しい表対象に前記の行情報を蓄える手段

を備えている請求項３１に記載の装置。

【請求項３５】　前記のブロックを読み取る手段は、前記のデーターベース表を含んでいるファイルを直接開く手段；そしてその開いたファイルから前記の表情報のブロックを直接読み取る手段を備え、

前記のブロックを読み取る手段は、前記のデーターベース表と組み合わされた管理システムから得られるＳＱＬインターフェースをバイ・パスする請求項３４

に記載の装置。

【発明の詳細な説明】

【０００１】

(発明の技術的分野)

本発明は、データベース・マネジメント・システム（ＤＢＭＳ）テーブルの再編成に関する。より厳密には、本発明は、ＤＢＭＳテーブルをユーザーが利用可能な状態に保ちながらＤＢＭＳテーブルを再編成することに関する。本発明は更に、ＤＢＭＳによるデータ検索のために設けられたＳＱＬインタフェースをバイパスしながら、ＤＢＭＳデータファイルから直接ＤＢＭＳテーブルをアンロードする再編成プロセスに関する。

【０００２】

(発明の背景)

最近のデータベース・マネジメント・システムには、益々、より大きなデータ記憶装置の保持が要求されてきている。データベースのサイズの増大に加えて、最近のデータベース内の構造は益々複雑になってきている。

【０００３】

通常、データベースはデータをテーブルの形で保持しており、各テーブルは、関連するデータの１つ又はそれ以上の行を保持している。例を挙げると、ある基本的なデータベース・テーブルは、例えば、組織に所属する個人の、名前、社会保証番号、住所、電話番号を有する複数の行を保持している。

【０００４】

データベースは、組織に新しいメンバーが追加されるにつれてサイズが大きくなり、メンバーに関する情報が追加されるにつれ、大きく且つ複雑になる。例えば、より大きくより複雑なデータベースであれば、上記情報に加えて、クラブメンバーの住所を示すマップを、おそらくはグラフィカルなフォーマットで保持することができるようになる。データベースは、勤務先住所、及び勤務先の場所を示す追加のグラフィカルなマップを追加すれば、更にサイズが大きく且つ複雑になる。

【０００５】

データベース・テーブルは、データベースの他のテーブル又は行へのポインタ

を含むようになれば、更に複雑になる。例えば、もう１つのテーブルに保持されている同僚のセットに対するポインタ、近隣組織のメンバーに対するポインタ、或いは、追加のメンバー・データが保持されているかもしれないあらゆる項目の番号に対するポインタなどである。

【０００６】

従来のデータベース・マネジメント・システム（ＤＢＭＳ）は、ブロックを割り当てることにより、データベース・テーブルを構築するためのスペースを提供する。テーブルが一旦定義されると、ＤＢＭＳは、関係するデータの行を記憶するのに必要なブロックを割り当てる。例えば、テーブルが１００，０００行を含むように構築され、テーブルの行が１ブロック当たり１００行入ると定義される場合、ＤＢＭＳはテーブルを構築するのに１，０００ブロックを割り当てる。

【０００７】

通常、ＤＢＭＳシステムは、ブロックを連続したブロックのセットとして割り当てる。割り当てられるブロックの連続したセットは、普通、エクステントと呼ばれる。一般的規則として、エクステントはそれぞれサイズが異なる。上記例を使えば、ＤＢＭＳは、１，０００ブロックのエクステント１つでも、５００ブロックのエクステント２つでも、要求される１，０００ブロックを割り当てることができるのであれば、どんなサイズのエクステントでも組み合わせてテーブルを構築するのに使うことができる。必要なブロックが割り当てられると、次に、割り当てられたブロックを使って、テーブル内の行にデータが記憶される。

【０００８】

何時でも、テーブルに追加データを付け加えることができ、ＤＢＭＳは要求に応じて追加のブロックを割り当てることになる。例えば、ユーザーが表に２５０行追加する場合、上記パラメータを使えば、３ブロックを追加して割り当てる必要があることになる。

【０００９】

又、何時でも、データベース内の情報を削除することができる。その場合、行の削除はＳＱＬの使用を通して行われ、テーブルから行が削除される。例えば、ユーザーは、ブロック１から５０行、ブロック２０から４０行、ブロック６０か

ら３０行削除することができる。そうなると、テーブル内には、テーブル内に存在するデータを保持するに必要なよりも多くのブロックがあることになる。

【００１０】

更に、データベース内のデータは更新される。例えば、上記データベース・テーブルで、まだ戦力に入っていない組織のメンバーが、その人の名前、住所、社会保証及び電話番号を保持する、割り当てられたテーブル内の行を持っているとする。戦力になると同時に行は更新され、職場住所と関連情報が入れられることになる。しかし、更新の結果として大量の情報が追加される場合、最初の行は、更新されたデータを保持するのには、割り当てられたデータのブロックでは十分でないかもしれない。

【００１１】

行が更新され、最初の行には更新された情報全てを保持するスペースが十分にない場合、行の移送が行われる。移送が行われると、行はより大きなスペースを備えた位置に動かされ、最初の行があったブロックには、移動（移送）された行の位置を指すポインタが置かれる。テーブルに対し大幅な更新があると、大量の行の移送が行われ、不適切な量のスペースが最初の行に割り当てられることになる。

【００１２】

しばしば、ブロック内に、更新されたデータを保持するのに十分なスペースの不足することがある。その場合には、行が、最初にあったブロックとは全く違うブロックに移送され、最初の行の位置には、異なるブロックの更新された行を指すポインタが置かれる。

【００１３】

行が移送されるときには常に、断片化と呼ばれる問題が起こる。断片化が起こると、ブロックを読み取るのに加えてポインタを読んで翻訳しなければならなくなるので、データベース情報の検索時間が大幅に増加する。行が他のブロックに移送されると、行特有の情報を検索する際には、少なくとも２つのブロック（ポインタを含むブロックと、移送され／断片化された行を含むブロック）を読まなければならなくなる。データベース・テーブル内の構造的変更も、断片化及び関

連する効率上の問題（例えば、行連鎖）を引き起こす。

【００１４】

時には、データベース・アドミニストレータ（ＤＢＡ）が、テーブルの状態に関する情報を提供する、ＤＢＭＳテーブルの分析を行う。例えばデータベース・アドミニストレータは、削除された行の数に関する情報を調べて、どれだけのブロックが削除された行を保有しているかに関する効率情報を確認する。或いは、データベース・アドミニストレータは、テーブルのどれだけの行が、他のプロセスによって移送され、又は断片化されてしまっているかを調べる。

【００１５】

断片化が沢山発生していれば、ブロックのサイズ及び行のスペースをもっと効率的に割り当てることができ、現在、テーブルのデータは効率的に検索されていないということになる。そのような場合、データベース・アドミニストレータはデータベースを再構築する決定を下すことが多い。

【００１６】

又別の例では、表を作る場合、ＤＢＡは自由ブロック比率（ＰＣＴＦＲＥＥ）又は使用ブロック比率（ＰＣＴＵＳＥＤ）を設定して、データベース・テーブルの構造に関する決定を行う。ＤＢＭＳは、各ブロックに行及びテーブル情報を書き込むので、自由ブロック比率を少なくともＰＣＴＦＲＥＥに等しくは保つことになる。

【００１７】

ＤＢＡは、ＰＣＴＦＲＥＥ変数を、データベース・テーブルがどのように使われることになるかによって設定する。例えば、テーブルが頻繁に更新されることになる場合、十分なスペースを利用できるようにして同じブロック内で必要な行の移送が全て行えるように、追加のＰＣＴＦＲＥＥを設定する。先に論じたように、同一ブロック内での行の移送は、テーブルを断片化することにはならない。移送されてはいるが断片化されてはいない行は、１つのブロックを読むだけで検索でき、移送された行が断片化されているときのように、あるブロックを読み、ポインタを翻訳し、第２（又はもっと多くの）のブロックを読む、というような面倒なプロセスで検索する必要はない。従って、ＰＣＴＦＲＥＥが適切に設定し

てあれば、データベース・テーブルが修正されても、ＤＢＭＳの性能は維持できる。

【００１８】

ＰＣＴＵＳＥＤは、ＤＢＡがＤＢＭＳテーブルの構造を制御できるようにするもう１つのパラメータである。ＤＢＭＳは、ブロックのパーセンテージがＰＣＴＵＳＥＤを下回らない限り、そのブロックに追加の行が配置されるのを妨げる。ＰＣＴＵＳＥＤは、更新がある場合にあるブロックが使われても、そのブロック内で使われているパーセンテージがＰＣＴＵＳＥＤを下回らない限り、新しい行を挿入するのには使われない、という意味でＰＣＴＦＲＥＥとは異なる。

【００１９】

随時、激しいＯＬＴＰアクティビティ（挿入、更新、削除）に巻き込まれているＤＢＭＳテーブルは、行移送、断片化、行連鎖などに見舞われることになる。更に、各種データベース・テーブルは、最初に構築されたときには、必ずしも適切な設定（ＰＣＴＦＲＥＥ、ＰＣＴＵＳＥＤ）になっているわけではなく、或いはデータベースの要件が変わったりして、テーブルに追加の移送、削除、断片化が生じることになる。そうすると、データ検索及びスペース利用の性能に低下を来すことになる。

【００２０】

ＤＢＡは、テーブルが効率的にデータを記憶しているか否かを判定するために分析を行う。その結果、ＤＢＭＳテーブルの１つ又はそれ以上が、データの記憶及び検索に効率的でないと判定されることになるかもしれない。テーブルの再編成（再構築）が、この問題に対する解である。最高の性能を発揮するには、テーブルを再構築（即ち、データを２次スペースにアンロードし、新しいテーブルを構築）する必要がある。このプロセスでは、断片化した行がアンロードされ、再構築されたテーブルに断片化無く記憶されることになるので、上記の望ましくない状況の多くが取り除かれる。

【００２１】

最近では、ＤＢＭＳテーブルを再編成する製品は、ＤＢＡが自己の環境内でデータベースをシャットダウンする必要なく、データベースをオンライン状態に保

ったままで、オンラインで行うように要求されている。しかしながら、再編成されているテーブル又はテーブルのリストは、再編成プロセスは対象となる単一又は複数のテーブルをロックするので、実際には、ユーザーには立入禁止となる。１週間に７日、１日に２４時間アクセスの必要なショップでは、これをやると許容できないダウン時間が発生することになりかねない。大きなオブジェクトをロックして再編成するのに必要な時間は、そのようなプロセスを現実的でないものとし、従って、ＤＢＡには再編成を行うのが不可能となる。

【００２２】

（発明の概要）

本発明の発明人は、再編成プロセスの間を通して、全ての意図された目的に対しデータベース・テーブルを利用可能な状態に保ったまま、データベース・テーブルをオンラインで再編成するという要件を理解している。

【００２３】

従って、本発明の目的は、再編成プロセスの間、単一又は複数のテーブルの行を完全に利用可能な状態に保ったままで、データベース・テーブルの再編成ができるようにすることである。

【００２４】

本発明の別の目的は、データベース・テーブルのオンライン再編成が行われる速度を上げる方法を提供することである。

【００２５】

上記並びにその他の目的は、ＳＣＮチェックポイントをソーステーブル上に確立する段階と、ＳＣＮチェックポイントの直前の状態にあるソーステーブルの再編成されたコピーを作る段階と、チェックポイント後に生じたトランザクションを再編成されたテーブルに適用する段階と、ソーステーブルを再編成されたテーブルに切り替える段階とから成る、データベース・ソーステーブルを再編成するための方法によって達成される。本方法は更に、トランザクションのエントリをトランザクション・テーブル内のソーステーブルに記録するためのトリガをソーステーブル上に作る段階と、トランザクションがソーステーブルで起こらないようにするためにソーステーブル上にロックを獲得する段階と、ソーステーブル上

にロックを獲得する前にトリガによって記録されたエントリを削除する段階と、ソーステーブルをアンロックする段階とを含んでいる。

【００２６】

トリガを作る段階と、ロックを獲得する段階とは、チェックポイントを確立する段階の前に実行される。ソーステーブルを再編成されたテーブルに切り替えた後、最初のソーステーブルはドロップして、新しい又はその他のテーブルがスペースを利用できるようにする。

【００２７】

本発明は、再編成プロセスの間、テーブルの行を完全に利用できる状態に保ちながら、ＤＢＭＳテーブルを再編成できるようにする方法及び装置を提供する。再編成プロセスは、１）トランザクションをトランザクション記録テーブルに記録するためのソーステーブル上のトリガ、２）ＳＱＬインタフェースをバイパスしてＤＢＭＳデータファイルから行を直接アンロードするための能力、という２つの主要な構成要素を有している。

【００２８】

再編成プロセスは、２つのフェーズで実行される。第１は、バルク・アンロード／ロード・フェーズであり、データの行がオラクルブロックからアンロードされる。対象のブロックには、時間内のある点までは触れてはならない。確立すべき時間内のある点とは、ソーステーブル上にトリガが作られる（即ち、トリガが確立された後に生じる全てのトランザクション及びブロック修正は、トランザクション記録に記録される）時点である。

【００２９】

再編成プロセスの第２のフェーズは、トランザクション記録テーブルを処理して、新しいテーブルをソースと同期させる段階である。新しいテーブルとソーステーブルが同期すると、再編成プロセスは、ソーステーブルをドロップし、新しいテーブルの名前をソーステーブルの最初の名前に変える段階を含む切替をプロセスに適用する。

【００３０】

（実施の形態）

データベーステーブルのオンライン再編成の概念理論は、テーブルをアンロックに維持してテーブルのユーザが利用できるようにしながら、テーブルの再編成を提供する。

【００３１】

再編成は、テーブルに対するＯＬＴＰアクティビティ（挿入、更新、及び削除）の結果である行連鎖、移行、及び一般的なデータ断片化を排除するプロセスとして定義されている。一般的には、再編成は、テーブルを物理的に再構築することによって遂行される。

【００３２】

本発明は、以下のことを想定している。

ａ）再編成プロセスは、ＤＢＡ特権である。

ｂ）再編成プロセスは、ソーステーブル＋再編成中に予測されるテーブルの成長を保持するのに十分に大きい使用可能な自由テーブル空間（ブロック）を有している。

ｃ）再編成中のソーステーブルは、再編成プロセス中にデータ定義言語（ＤＤＬ）を受けることはない。

ｄ）ソーステーブルは、通常のＯＴＰＬアクティビティのために使用可能である。

ｅ）再編成プロセスは、通常の応用がＯＴＰＬ環境においてそのようにする必要があるのと全く同じように、テーブルのロックに関して時間から時間へ競合する。

【００３３】

この点において、本発明が一般的な用語で、且つオラクルデータベースシステムの構造と矛盾しない特定例を使用して説明されることに注目することが重要である。しかしながら、完全に異なる構造を使用する他のデータベースシステムも本発明の範囲内にある。

【００３４】

例えば、データベーステーブルを再編成するためには、データベーステーブルの構造を知らなければならない。オラクルテーブルの構造内のキー要素は、“ｒ

ｏｗｉｄ”としても知られているオラクル行アドレスである。

【００３５】

ｒｏｗｉｄは、ファイル番号、ブロック番号、及びスロット番号からなる。これは、オラクル行を独自に定義し、探知するために使用することができる。ファイルはブロックを含み、ブロックはスロットを含み、スロットはブロック内のオラクル行を指し示す。しかしながら、ページまたは他の編成技術をベースとするデータベース構造も適用する。この場合、特定の行情報は、他のデータベース構造と矛盾しない関連ページ及び行識別子を識別することによって入手する。

【００３６】

別の例として、ＤＢＭＳは一般的に、データベース内に発生する変化（ＯＬＴＰアクティビティ）をレコードする、またはログする能力を提供する。オラクルＤＢＭＳでは、データベースの状態の全てのトランザクションまたは変化がシステム変化数（ＳＣＮ）に対してレコードされる。オラクルＤＢＭＳは、ＳＣＮの独自性及びプロパティの不断の向上を保証する。従って、データベースの変化を識別した時にはＳＣＮが使用され、また本発明を別のＤＢＭＳに適用する時には別の同じようなプロパティが使用される。

【００３７】

さて図面を参照する。幾つかの図面においては、同一の、または対応する部品に対しては同一の参照番号を付してある。図１Ａは、本発明によるオンライン再編成プロセスの高レベルプロセスを示す流れ図である。再編成はステップ１００から開始され、再編成プロセスは再編成入力パラメタを受ける。再編成入力パラメタは、再編成されるテーブルの名前のようなアイテム、及び再編成されるテーブルをどのように統合すべきかに関する特定の情報を与える他のパラメタを含んでいる。表１は、本発明により使用される入力パラメタのリストである。例えば、限定するものではないが、PCTFREE及びPCTUSEDを含む他のパラメタも入力であることができる。

【００３８】

表　１

ユーザが定義できる再編成パラメタの定義。入力は再編成プロセスに使用可能

にされる。

Login	ユーザのログイン名
Passwird	ユーザのパスワード
Sid	オラクルＳｉｄ
Home	オラクルホーム
DoFailurePrediction	ドゥ故障予測フラグ
FirstLock.Wait	初期ロック待機（秒）
FirstLoc.Retry	再試行の初期ロック数
Table.Name	テーブル名
Table.Owner	テーブル所有者
Table.ChangeDDL	適用するために新テーブル変更
WorkSpace	一時的に格納するためのテーブル空間名
ExportDirectory	ファイル格納のために使用されるファイルシステムディレクトリ名
Tablespaces	新テーブル格納情報
TransactionLock.Wait	トランザクション適用中のロックパラメタ
TransactionLock.Retry	
WindowTimeOut	テーブルをロックし続けることを許容される時間
Switch	スイッチ戦略情報
Threshold.Tries	新テーブルへのスイッチング試行時間なし
Threshold.Exceed	もしThreshold.Triesが超過すれば、アクションのインディケータ

【００３９】

ステップ１００に含まれているのは初期化プロセス（図２参照）であって、データベースへの接続（オラクル、この例ではステップ２００において）、予備セットアップの遂行（ステップ２１０）、データベースが再編成される状態にあることを確認する事前編成チェックを遂行する故障予測プロセス２５０の呼出しを含む。

ステップ１０５において、要求に応じていろいろな初期化が遂行される。例え

ば、トランザクションを格納するためのログファイル（以下、トランザクションテーブルという）が作成される。表２は、トランザクションテーブルの定義例であって、詳細に関しては後述する。

【００４０】

表　2

```
CREATE TABLE <c.owner>. TSREORG$LOG(\
                    ROW$$ rowid\
                    dmltype char(1). \
                    updtime date) <storage parameters>
```

ステップ１１０において、ソーステーブル上で遂行される何等かのアクティビティ（ＯＬＴＰ情報を含む）をログするソーステーブル上にトリガが作成される。トリガは、例えばｒｏｗｉｄ、トランザクションの型、及び現タイムスタンプのようなログ情報を入力する。トランザクションテーブルは、“Ｉ”即ち挿入、更新のための“Ｕ”、及び削除のための“Ｄ”のような参照を含むトランザクションの型のための識別子を含む。ソーステーブルに対して発生するトランザクションを識別するための他の方法（例えば、トランザクションを識別するナンバーコード）も使用することができる。表３は、トランザクションテーブルと共に動作するように設計されたトリガを実現するためのプログラミング構造の例を示している。

【００４１】

表　3

```
CREATE OR REPLACE TRIGGER <c.owner>. TSREORG$TRIG
        after insert or update or
        delete on <c.owner.c.table>\
        referencing old as old\
        for each row    declare dmltype char; \
              begin     if inserting then
                          dmltype:='I'; \
                        elsif updating then dmltype:='U'; \
```

```
elsif deleting then dmltype:='D'; \
end if; \
insert into <c.owner>. TSREORG$LOG
    (row$$, dmltype)
```

図３は、トランザクションテーブルの作成及びトリガがステップ１１０に示されているように組合わされる本発明の一実施の形態を示している。図３において、トランザクションテーブルが作成され（ステップ３００）、ユーザ定義一時テーブル空間３２０内に格納される。ユーザ定義一時テーブル空間３２０は、ソーステーブルへの何等かの予測される追加及び更新を保持するのに十分な格納空間を提供する。

【００４２】

トリガ及びトランザクションテーブルがセットアップされると、ソーステーブルはロックされ、チェックポイントが作成される（ステップ１１５）。ソーステーブルに対するロックは、テーブルに対して更新が遂行されるのを防ぐ何等かのプロセスを介して取得することができる。一般的に言えば、ロックを取得するための最初の試みに成功しなければ、排他的なロックを取得するための待機時間の長さ及び試行数が適用される。

【００４３】

図４は、排他的ロックを取得し、チェックポイントを取得するためのプロセスの詳細を示している。ステップ４１０において、排他的ロックが要求される。もしロックが入手されなければ、再試行プロセス４２０が呼出される。ソーステーブルがロックされると、チェックポイントのためのＳＣＮ数が入手される（ステップ４３０及び４４０）。

【００４４】

チェックポイントは、時間内の、またはデータベースの状態内のある点を独自に識別する何等かの識別子、または他の方法（例えば、タイムスタンプ、チェックポイントと別の参照との組合わせ、または外部参照のような）である。例えば、オラクルデータベースシステムでは、データベース内で発生する各トランザクションに対してＳＣＮが適用される。オラクルシステムはＳＣＮの独自性及び不

断の品質向上を保証するから、それはチェックポイントとして使用するのに十分に適している。従ってステップ１１５において、ソーステーブルがロックされ、ＳＣＮまたは他の類似識別子（CHECKPOINT_SCNまたはReorgScnとも呼ばれる）がチェックポイントとして入手される。表４は、ReorgScnを含む再編成プロセスをプログラミングするのに使用される変数のリスト及び説明である。

【００４５】

表　4

再編成プロセス中に使用される応用変数の説明

ReorgScn	再編成の定義されたシステム変化回数
redoFiles	応用定義されたレドゥー情報ファイル
undoFiles	応用定義されたアンドゥー情報ファイル
ddlFiles	応用定義されたＤＤＬ情報ファイル
insertSimt	テーブルロードのための挿入ステートメントＤＤＬ

ステップ１２０において、ソーステーブルがロックされている間に、ソーステーブルのセグメントヘッダーが読出され、ブロックのリスト（エクステントリストとも呼ばれる）が作成される。ソーステーブルのセグメントヘッダーは、ソーステーブルによって使用するために割当てられたデータブロックを識別する識別子を格納するファイルである。エクステントリストは、ソーステーブルデータがアンロードされるブロックを識別する。

【００４６】

オラクルデータベース管理システムでは、セグメントヘッダーは、テーブルが作成された時に、またはそのテーブルのために既に割当て済みの空間の他に付加的な空間を必要としたテーブルの更新中に割当てられるエクステント（隣接するデータのブロック）を識別する。従って、オラクルシステムでは、ステップ１２０において作成されるエクステントリストは、基本的には、ソーステーブルのために割当てられた各エクステントを含み、これらの各エクステントはテーブルデータを格納する隣接するデータのブロックのセットを識別する。

【００４７】

本発明に関して言えば、ソーステーブルのセグメントヘッダーは、ソーステー

ブル内に格納されているデータが何処に位置しているかのリストを維持している何等かのファイルまたは記憶デバイスとして広義に定義されている。例えば、データベース管理システムは、ページに基づいてデータ空間を割当てることができる。この場合、ステップ１２０において作成されるエクステントリストは、ページ情報、及びデータがソーステーブルの何処に格納されているかを識別するために必要な他のデータを含む。従って、どのような数のデータベース管理システム及びそれらの個々の構造も、本発明の教示を使用すれば受入れ可能である。

【００４８】

ステップ１２５において、ソーステーブルがアンロックされる。ソーステーブルのアンロックに先立って、ステップ１１５においてチェックポイントを確立する前にトランザクションテーブル内において生成された行が削除される（例えば、図４のステップ４５０）。トリガを作成し、テーブルをロックし、チェックポイントを確立し、そしてチェックポイントの確立の前にトランザクションテーブル内にログされたトランザクションを削除するプロセス（チェックポイントを確立する前にテーブルがロックされているから、このプロセスはソーステーブルがロックされている間にトランザクション内の全てのエントリを削除することによって遂行される）は、テーブルがロックされ且つチェックポイントが確立された後に発生したエントリだけを維持することを保証する。

【００４９】

図１Ｂを参照する。ステップ１３０において、ソーステーブルのコピーが作成される（以下、新テーブルという）。新テーブルは、初めに作成されたソーステーブルに基づいており、テーブルに対する全ての適用可能なユーザが要求したＤＤＬ変更を含んでいる。

【００５０】

図５は、ソーステーブルの生成を示すフローチャートである。ステップ５００において、ソーステーブルを定義するデータ定義言語（ＤＤＬ）計画が検索される。ＤＤＬの変更を要求されたユーザはＤＤＬ（５１０）に適用され、索引を含む新しいテーブルオブジェクトが生成される（５２０）。これらのプロセスは初期設定で生成された再生及び取消ファイルに記録される（ステップ１０５、図１

Ａ参照）。新しいテーブルをロードするための挿入命令文は後のレファレンス（５４０）のために記憶される新しいテーブルのＤＤＬ（５３０）に基づいて生成される。

【００５１】

結局、行データはソーステーブルからアンロードされ、新しいテーブルにロードされるであろう。ステップ１３５において、マッピングテーブルが作られ、ソーステーブルのｒｏｗｉｄを新しいテーブルに挿入された行のｒｏｗｉｄにマップする。ＤＢＭＳが利用可能なｒｏｗｉｄを作るかどうかに拘らず、マッピングテーブルは、行がソーステーブルからアンロードされると共にそれらが新しいテーブルに記憶されることに関する完全な情報を含むように設定される。

【００５２】

ステップ１４０において、データ記憶装置（アンロードされたブロック範囲）は、ソーステーブルからアンロードされたブロックのファイル番号及びブロック番号の範囲を記憶するために作られている。本発明はデータ記憶装置にアンロードされたファイル／ブロック番号のレコードを作る。オラクルは隣接範囲にブロックを配置するので、アンロードプロセスは隣接範囲、したがって、オラクルシステムにおいて、ブロックをアンロードしそうであり、ファイル／ブロック番号の組合せの範囲を記憶するのに十分であろう。これは、アンロードされた全てのファイル／ブロック番号を記憶するのとは対照的に、記憶装置の要求を最小にする利点を有している。しかし、別のＤＢＭＳにおいて、それぞれの個々のファイル／ブロック番号の組合せの記憶装置又はアンロードされたブロックをトラッキングする他の方法はより有効なことがある。

【００５３】

ステップ１４５において、データブロックはソーステーブルからアンロードされ、そこに含まれたデータは新しいテーブルのブロックにロードされる。アンロードプロセスは、ステップ１２０で作られたエクステントリストからデータブロックを読むことを含んでいる。その後、行はアンロードブロックから抽出され、新しいテーブルにロードされる。

【００５４】

速度を改善するため、再編成プロセスはＤＢＭＳのＳＱＬインターフェースをバイパスし、データブロックを読み込み、ＤＢＭＳデータファイルから直接行のデータを抽出する。ＤＢＭＳのＳＱＬインターフェースをバイパスすることは、ＤＢＭＳファイルの構造に関する情報を要求する。ＤＢＭＳファイルに記憶されたデータブロックからのデータの行を検索することはブロックの構造の知識を要求する。ＤＢＭＳブロック及びファイルの構造は、構造を説明する仕様書、又は既に作られたブロックの調査のいずれかにより得られてもよい。

【００５５】

例えば、図６は、オラクルデータブロックの構造を示している。オラクルテーブルの各データブロックは、ブロックヘッダー６００、ブロックトランザクション情報領域６１０、及び行データ６３０へのポインターを含むテーブル及び行のディレクトリ６２０からできている。オラクルでは、行データそれ自体はボトムアップで満たされている。

【００５６】

上述した構造情報を利用して、オラクルＤＢＭＳファイルは開かれてもよく、データブロックが読まれ、行情報がそこから抽出される。抽出された行情報は新しいテーブルにロードされる。上述したように、同様の操作がオラクル以外のＤＢＭＳシステムで行われてもよく、例としてここに示されている。

【００５７】

アンロードプロセスは、ステップ１１５で確立されたＳＣＮ番号（ＣＨＥＣＫＰＯＩＴ＿ＳＣＮ）を使用する。ＣＨＥＣＫＰＯＩＴ＿ＳＣＮ以下のＳＣＮを有するブロックだけがアンロードされるだろう。ＣＨＥＣＫＰＯＩＴ＿ＳＣＮ以上のＳＣＮを有するブロックはスキップされるだろう。スキップされたブロックリストもまた、アンロード／ロードプロセスでスキップされるブロックを認識するために維持される。

【００５８】

ステップ１４５のアンロード／ロードブロックは図７により詳細に示されている。ステップ７００において、エクステントリストが読まれ、リストの各ブロックのＳＣＮが決定される（７２０）。

【００５９】

ブロックＳＣＮがＣＨＥＣＫＰＯＩＴ＿ＳＣＮ以下の場合（ＲｅｏｒｇＳｃｎ）には、チェックポイントが確立されるから、そのブロックが変更されないことを示している。この場合、そのブロックはソーステーブルからアンロードされて新テーブル内へロードされるように指令され（ステップ７５０）、そのブロックはアンロードされたブロック範囲へ追加される（７６０）。もしブロックＳＣＮがCHECKPOINT_SCNより大きければ、そのブロックはスキップされたブロックのリストに追加される（ステップ７４０）。エクステントリスト内の各ブロックは、同一プロセスに適用される。

【００６０】

各行がアンロードされて新テーブル内へ挿入されると、ソーステーブルのｒｏｗｉｄが新テーブル内のその新ｒｏｗｉｄへマップされ、マッピングテーブル内に格納される。マッピングテーブルは、もし必要ならば、再編成プロセスの後半において特定の行の削除に使用される。マッピングは、ＤＢＭＳ内のテーブルの形状で、メモリ内に、ファイルシステム上に、またはマッピングテーブルを維持して後刻再編成プロセスが使用できるようにする他の何等かの方法で格納することができる。表５は、本発明の一実施の形態において使用されるリストを説明している。

【００６１】

テーブル　５

再編成処理の間に使用される記述リスト

前記リストにおけるエクステント・リスト・ノードが含むもの

ＦｉｌｅＮｏ	オラクル・データ・ファイル番号
ＢｌｏｃｋＮｏ	開始オラクル・データ・ブロック
Ｌｅｎｇｔｈ	このエクステントにおけるブロックの数

前記リストにおけるトランザクション・ブロック・リスト・ノードが含むもの

ＦｉｌｅＮｏ	オラクル・データ・ファイル番号
ＢｌｏｃｋＮｏ	オラクル・ブロック番号
Ｃｏｕｎｔ	このブロックに関するトランザクションの数

前記リストにおけるトランザクション・リスト・ノードが含むもの

ｒｏｗｉｄ　　オラクル列アドレス

ｔｙｐｅ　　Ｉ＝挿入、Ｕ＝更新、Ｄ＝削除のいずれかであるトランザクションの種類

前記リストにおける削除及び挿入リスト・ノードが含むもの

ＳｌｏｔＮｏ　　列がオラクル・ブロックに記憶されるスロット

【００６２】

テーブル５に記載のとおり、前記エクステント・リストは、ソース・テーブルをアンロードする処理の間、ブロック情報を識別するために利用される。トランザクション・ブロック・リストは、発生したトランザクションのブロック及び種類を識別する。トランザクション・リストは、トランザクション・ブロック・リストにおいて識別された各トランザクションに関して、ｒｏｗｉｄ及びトランザクションの種類を供給する。そして最後に、削除及び挿入リストは、トランザクション・リストにおいて識別された各ｒｏｗｉｄに関してスロット番号を識別する。上記リストは、トランザクション・テーブルから情報を検索することによってコンパイルされ、及びアプライ・トランザクション処理（例えば、図９参照、ブロック９１０、及び９２０）において利用される。

【００６３】

図８は、本発明によって利用される別個のアンロード及びロード処理（スレッド）を示す。ロード・スレッドは、アンロードすべき特定のブロックを示すメッセージ８０５によって開始する（例えば、図７参照、ステップ７５０）。

【００６４】

各列が抽出されると（ステップ８２０）、それは共有列転送領域８４０へと転送され、アンロードされた列が、転送の準備ができていることを示すメッセージが、ロード・スレッドに送信される（ステップ８５０）。

【００６５】

以前に初期化された（ステップ８５５）ロード・スレッドは、転送の準備ができている列を示すメッセージを受信する。転送領域における列が検索され、及び前記列からのデータは、新しいテーブルへとロードされる（８７０）。列のアン

ロード、転送、及び新しいテーブルへのデータのロードは、ブロックにおける各列に関して、アンロード・スレッド及びロード・スレッドによって反復される（ステップ８９０及び８９５）。

【００６６】

アンロード及びロード処理の持続期間に関して、ソース・テーブルは、通常のＯＬＴＰ活動に関してユーザが利用可能である。列上で作られたトランザクションは、トリガを経由して、トランザクション・テーブルに記録される。再編成処理の第二段階は、記録されたトランザクションを処理することである（アプライ・トランザクション処理、ステップ１５０）。

【００６７】

後述のアプライ・トランザクション処理は、再編成が発生した／発生している間に、ソース・テーブルに生じたトランザクションに従って、新しいテーブルを更新し、及び図９Ａ及び９Ｂにより詳細に記載されている。第一のステップは、ソース・テーブルをロック（lock）し（図９Ａ、ステップ９００）、及びファイル／ブロック番号によって編成されたトランザクションを識別するトランザクション・ブロック・リスト９２０を作成する（ステップ９１０）ことである。これをすることの効果は、前記処理が、個別のトランザクションを処理するよりむしろ、一度に一つのブロックで生じたトランザクションを適用することができることである。

【００６８】

トランザクション・ブロック・リストが空である場合、スキップされた（skipped）ブロック・リストに配置されたブロックは、アンロード／ロードされる（ステップ９３０）。テーブルがロックされたのでトランザクションが未完のままである場合、ソース・テーブルは、トランザクションを発生させるためにアンロックされ（９４０）、及びアプライ・トランザクション処理が開始する。

【００６９】

図９Ｂへと続いて、トランザクション・テーブルにおいて見られる一定のファイル／ブロックに関して、このファイル／ブロック結合がアンロード処理に関与したか、決定がなされなければならない（ステップ９５０）。このチェックは、

アンロード／ロード・ブロック処理７５０の間に、以前に記憶されたファイル／ブロック範囲（７６０参照）に対してなされる。ファイル／ブロック結合が見つからなかった場合、このブロックの列は、新しいテーブルへとアンロード及びロードされるだけでよい（９６０）。これは効果的に、このブロックに関して生じたすべてのトランザクションを処理する。さらに、ファイル／ブロック結合エントリは、スキップされたブロック・リストから取り除かれ、アンロードされたブロックのファイル／ブロック範囲へと追加される。

【００７０】

ブロックが見つかった場合とは、このファイル／ブロック結合における列が、すでに新しいテーブルに存在することを意味する。一つの実施形態において、本発明は、以前にアンロード／ロードされたブロック上でトランザクションを実行するために、ブロック処理ごとに処理トランザクションを利用する（９７０、図９Ｂ）。

【００７１】

ブロック処理ごとの処理トランザクションは、図１０Ａに記載されている。第一に、トランザクション・テーブルは、特定されたブロックに関するトランザクションを有する列を決定するために利用される（ファイル番号、ブロック番号結合、ステップ１０１０）。それから、存在するのであれば、削除及び挿入に関するスロット番号で、リストが作成される（ステップ１０２０／１０３０）。更新は、削除及び挿入として扱われるであろう。

【００７２】

リストは、意味のあるトランザクションのみが実行される必要がある方法で、作成される。例えば、一定のスロットに関して、挿入／削除／挿入が識別される場合、挿入のみが必要とされるであろう。削除に関しては、削除されたソース・テーブル列のｒｏｗｉｄに基づいて、新しいテーブルから削除されるべき列を決定するために、マッピング・テーブルが使用されるであろう。挿入に関しては、ブロックは以下の制約条件を伴ってアンロード及びロードされる：挿入リストに存在する列のみが、アンロードされる必要がある。挿入リストは、一定のファイル／ブロック番号に関する、スロット番号のリストである。

【００７３】

ここで図１０Ｂを参照すると、前記リストが作成された後、削除リストに項目が存在する場合、それらは（例えば、ｒｏｗｉｄ又はスロット番号を増加することによって）編成され（ステップ１０５０）、それから新しいテーブルから削除される（ステップ１０５５）。同様の方法で、項目が挿入リストに存在する場合、それらは編成される（ステップ１０６０）が、新しいテーブルに挿入される（ステップ１０６５）。前記挿入処理は、上述のアンロード／ロード処理７５０を含む。

【００７４】

他の実施形態において、前記トランザクションは、このファイル／ブロック結合に関して挿入されたすべての列を、新しいテーブルから削除することによって処理され、それから前記ブロックは、ソース・テーブルからリロードされる（９８０）。

【００７５】

トランザクションが処理された後、トランザクション・テーブルにおいて対応する列は、トランザクションが二回処理されるのを防止するために、削除される。ブロック処理ごとの処理トランザクションは、処理すべきトランザクションがなくなるまで続く。

【００７６】

ソース・テーブルはオンラインなので、トランザクション・ログ・テーブルは、トランザクションが処理されるよりも迅速に、満たされることが可能である。追加の制御パラメータが、この状態を制御するために利用されてもよい。例えば一定のしきい値（前記処理が実行された回数）の後、前記アクションが、前記テーブルをロックし及び残りのトランザクションを消費するか、又は再編成を含むものであってもよい。

【００７７】

他のパラメータが、トランザクションを処理するために利用可能性のあるウィンドウを制御するために利用されてもよい。これは、ソース・テーブルにアクセスするすべての他のユーザと、再編成処理にアクセスするすべての他のユーザと

の間での時間の共有を可能にするであろう。

【００７８】

データ・ブロックを更新し又は変更する処理の間、データベース管理システムは、必ずしも前記更新又は変更のすべてを、即座にディスクに書き込まず、結果としてダーティな（dirty）ブロック（更新されたが、まだディスクに記憶されていないブロック）を生む。効率のため、ブロックは時々メモリにキャッシュされ、それから、個別の書き込みが各更新又は変更に関して処理するよりも、いくつかのブロックが、単一のディスク書き込み処理において、ディスクに書き込まれてもよい。

【００７９】

オラクル（Oracle）において、システム・チェックポイント（system checkpoint）は、すべてのダーティなブロックが、持続性記憶装置（ディスク）に書き込まれることを確実にする。このシステム・チェックポイントは、ＤＢＭＳによって自発的に呼び出されるか、又はプログラミング（例えば、本発明のオンライン再編成処理）あるいはＤＢＡによって明確に要求されてもよい。

【００８０】

（ディスクに書き込まれ又は書き込まれなかった）トランザクションのリストを処理するためのデフォルトの方法は、第一にシステム・チェックポイントを適用し又は呼び出すことで開始する。しかしながら、システム・チェックポイントは、時間がかかり、最終のトランザクション及び最終のシステム・チェックポイントの発生回数のチェックは、前記ステップが必要であるかを明らかにする。最終のトランザクション時が、最近のシステムのチェックポイント時の前に発生した場合、システム・チェックポイントは必要とされないであろう。

【００８１】

例えば、最終のトランザクションが１１時に発生した場合、最終のシステム・チェックポイントは１１時１０分に発生し、及び未完のトランザクションのリストが処理される時間は１１時２０分であり、追加のシステム・チェックポイントは、必ずしもこの時点ではない。反対に、最終のトランザクションが１１時１５分だった場合、又は最終システム・チェックポイントにおけるいつか又はその後

に、追加のシステム・チェックポイントは、すべてのトランザクションがディスクに書き込まれたことを確実にするために必要とされるであろう。

【００８２】

さらなるエントリがトランザクション・テーブルに残らなくなると、初期アンロード中のスキッピング・ブロックの間、維持されていた、スキップされたブロック・リストがチェックされる。アプライ・トランザクション処理の間に処理されたブロックは、スキップされたブロック・リストから取り除かれる／取り除かれた。図９Ａにおいて、スキップされたブロック・リストに残っているファイル／ブロック・エントリは、対応するファイル／ブロックから新しいテーブルへ、列をアンロード／ロードすることによって処理される（ステップ９３０）。

【００８３】

すべてのトランザクションが処理されると、新しいテーブルは、ソース・テーブルを取り替え（スイッチ・ソース・テーブル処理、ステップ１５５）、及びクリーンアップ（cleanup）処理が呼び出され（ステップ１６０）、再編成処理を完了させる。これは、トランザクション・ログ・テーブルに残っているエントリがなく、及びスキップされたブロック・リストが空である時に、さらなるトランザクションが発生するのを防止するために、ソース・テーブル上で最終ロックを得ることによって、実行される。再編成処理は、ソース・テーブルがドロップ（dropped）されうるか、及び（最新の変更で更新されている）新しいテーブルが、前記ソースへと再命名されうるかを決定しなければならない。再編成処理は、（１）現在、ソース・テーブルへと読み取り専用アクセス（read only access）のみを有するユーザと；及び（２）テーブルがアンロックされるのを待っている未完のトランザクションの待ち行列に加わったユーザとの両方を考慮する。

【００８４】

第一のケースにおいて、以下のアクションが取られうる（制御パラメータＮＩＣＥに基づく、ステップ１１００、図１１）：（１）テーブルがドロップされるように、読み取りアクセスに関与するユーザ・セッションをキル（kill）する（ステップ１１１０）；又は（２）すべてのユーザ・トランザクションが終了するまで、ある時間間隔でループする（１１５０）。この時点で、テーブルは交

換される（ステップ１１２０）。ステップ１１２０は、依存したオブジェクトの作成（インデックス、外部キー、ビュー、等）、新しいテーブルのリリース（名前は、ソース・テーブル名に変更される）を含み、前記ソース・テーブルはドロップされるか、又は再命名され（バックアップの目的のために保管する）、再編成処理を完了させる。新しいテーブルはロックされていないので、再命名されるとすぐに、ＤＢＭＳのユーザはそれを利用することができる。

【００８５】

第二のケースにおいて、トランザクションが未完である時、再編成処理は、前記ロックを解放し、上述のアプライ・トランザクション処理を繰り返す。

【００８６】

テーブル６は、ここで検討されている再編成処理の予備的な原型である。前記予備原型は、コンパイル可能なもしくは実行可能なプログラムもしくはプログラム設計言語でもなく、ここで検討されているすべての特性を含むものでもないが、上述の発明の原理と一致した、オンライン再編成において実行される様々な機能及び処理を示すものである。

【００８７】

TABLE 6

Online Reorg prototype design.

1. Parse and store the control file parameters.
Steps.
1. Use lex and yacc to parse the file.
Implementation. modify the fao_yacc.y and fao_cmdline.1 files in
The Fast Analyzer source to parse the control file.

【００８８】

```
                    For options supported in the control file see
orrg_interface.txt.
                    On an option not specified use default.
                    Errors.
                         Exit on any option defined as needed and
not present in
                        control file.
               2. Map values obtained to structure of
                     control.login = <login name value>
                     control.passwd = <login password value>
                     control.sid = < oracle sid value>
                     control.home = < oracle home value>
                     control.table = < table name value >
                     control.owner = < table owner value>
                     control.commitSize < commit size value>
                     control.iLock = < NOLOCK|TIMEOUT|FOREVER>
                     control.iLockWait = < No. of seconds value>
                        control.iLockRetry = < No. of time to try
value>
                     control.atLock = < TIMEOUT|FOREVER value >
                     control.atLockwait    = < No. of seconds
value>
                      control.atLockRetry      = < No. of times to
retry>
                        control.atwindow    = < Window for applying
transactions secs>
                                          control.switch          = <
IMMEDIATE|AFTER_ALL_TRANS|NICE value>
                  NOTE future refrences to control structure will
alias to c.
                    e.g. control.login = c.login
2. Connect to oracle.
          Steps.
           1. Use ora connect in fast analyze api pass c.login ,
c.passwd and c.sid
                  see connect.h for complete list of parameters.
             Errors.
                       Exit if error received from ora_connect.
3. Do setup of table specified for the online reorg.
          Steps.
                1. If c.iLock <> NOLOCK then Lock table using
c.iLockwait, c.iLockRetry, c.iLock.
                    Error. on error or timeout exit.
          2. Check point system.
                    SQL= "alter system checkpoint global"
          3. Find system change number for the checkpoint.
                    SQL= "select checkpoint change# from v$database"
                    REORG SCN=<result>
          4. Create trigger and transaction table to keep an audit
of all future transactions.
                    Create transaction table as TSREORG$LOG
```

【００８９】

```
            SQL= "create sequence <c.owner>.TSREORG$LOGID
increment by 1";
            SQL= "create table <c.owner>TSREORG$LOG ( \
                        logid number,
                        M ROW$$ varchar(255), \
                        dmltype char(1)  \
                        updtime date \
                              )"
         5.   Create trigger on TABLE(c.owner,c.table)
                              SQL="CREATE  OR  REPLACE  TRIGGER
<c.owner>.TSREORG$TRIG
                           after insert or update or delete on
<c.owner.c.table> \
                        referencing old as old \
                        for each row \
                             declare dmltype char; \
                             begin if inserting then dmltype:='I' '
\
                                elsif     updating  then dmltype
'U';      \
                                 elsif     deleting  then dmltype
'D';      \
                        end if;
                        insert into <c.owner>.TSREORG$LOG
                (TSREORG$LOGID.next,m_row$$, dmltype)"
          6. commit 4 and 5 this will unlock the table if locked.



               Error. On error exit.
               Implementation.
                    Write a function 'create$ql' that will take a
 sql stmt and form a new state
     ment replacing c.owner with value and c.table with value.
                   Write a function 'exec$ql' that will exec a sql
 statement.
                        These functions should be used in steps
 1,2,3,4,5.
                   commit will use oci ocom.
     4. Create new table and associated DDL.
          Steps.
          1. Create new table
               SQL='create table c.owner, c.TSREORG$TEMP
                    as select * from <c.owner>.<c.table>
                    where 1 = 2'
          This should create an empty table.

          2. Create a table to map old rowids and new rowids.

          SQL='create table <c.owner>.TSREORG$MAP
                    file#number,
```

【００９０】

```
                block#number,
                slot#number, nrow$$
                varchar2(18) )"
        3. Create indexes for map table.
                 SQL="create index <c.owner>.TSREORG$IDX_1 on
<c.owner>.TSREORG$MAP (1
                      (orow$$)'
                SQL="create index <c.owner>.TSREORG$IDX_2 on
<c.owner>.TSREORG$MAP(\
                (nrow$$)"

        4. commit the above.
        implementations .
            Again createSql and execSql can be used to create and
exec the sql.
    8. Unload and load rows from blocks where BLOCK SCN < REORG
SCN defined in step 3.
        Steps.
        1. read extent list.
        2. get columns of <c.owner.table> and create column list.
        3. from column list create insert sql.
              SQL= insert into <c.owner>.TSREORG$TEMP values (
i1,...in).
        5. allocate memory to such that memory >= no columns*max
column size*array size.
        6. unload rows from a block into memory.
        7. load rows from memory using oci into TSREORG$TEMP.
                    for each row to be loaded hold on to
fileno,blockno,slotno of the row
             SQL=declare nrowid rowid ; begin
                    insert into <c.owner>.TSREORG$TEMP values
(i1,...in);
                nrowid = DBMS_SQL.LAST_ROW_ID();
                    insert into <c.owner>.TSREORG$MAP values
(fileno,blockno,slotno);
                end;
         8. commit based on c.arrSize. The above sql can be done
using array inserts.
              9. store in memory every block unloaded as (
fileNo,BlockNo) in BlockHeap.
          Implementation. Use modified fast unload source and
tsreorg insert source
             to achieve 1,8.
                Create module to efficiently insert,delete and
retrieve
                 fileNo,BlockNo values into BlockHeap.
            Implementation of BlockHeap. The BlockHeap will be
constructed as a tree
        using the otree api already developed. the BlockHeap Node
        will contain the following elements.
             BlockHeap Node FileNo
                           Start
```

【００９１】

Len.
The insert into a tree will be modified as following.
For a new n.FileNo,n.Block the following modifications must be made to
accomodate range values.
if n.FileNo != c.FileNo where c is the current node being examined.
then insert as per tree insert.
if n.FileNo == c.FileNo && n.Block != c.Start
if n.Block is in (c.Start+c.len) then c.len++
else insert new node as per tree insert
set new.Start=n.Block new.FileNo=n.FileNo and new.len=1
if n.FileNo == c.FileNo && n.Block == n.Start do nothing.

Search for a s.fileNo,s.Block in the BlockHeap will be conducted as follows.
Found=False.
do until Found=TRUE OR end of tree is reached. Then Found =False.
if (s. fileNo, s. Block == c. fileNo,c. Start) then Found=TRUE return.
if (s.fileNo == c.fileNo && s.Block != c.Start)
if (s.Block in (c.start + c.len) then Found=TRUE return.
if (end of tree) return Found=FALSE.
else continue to next node.

9. Loop application of transactions that have occured and stored in TSREORG$LOG.
Steps.
1. Lock <c.owner>.<c.table> in read only mode. using c.atLock, c.atLockWait,
c.atLockRetry.
2. After Lock obtained proceed to select a list of transactions to process.
2. LastUpdateTime=select max(updtime) from TSREORG$LOG;
3. LastCheckPointTime=select check_point from V$THREAD;
4. if LastCheckPointTime < LastUpdateTime
apply system checkpoint.
5. Select blocks for appltying transaction updates to the new table.
select FILE(mrow$$) , BLOCK(mrow$$),updtime
from TSREORG$LOG where updtime > LastAppliedTime
order by 1,2,3 intial LastAppliedTime is NEVER.
6. Taking a Block(file,block) at a time. Check Block Heap if
Block is present i.e already unloaded.
7. Start Window Timer based on c.atLockWindow.
8. if (not Found) then
8. Unload this block

【００９２】

```
                9. Insert unloaded rows into new table.
                10. update BlockHeap, with new block
             else
                select transaction list for this block.
               foreach row ( File,Block,Slot) update a slot_bit
array
                  in the form slot bit[ slot ] = 0x01(binary
00000001) for insert
                        slot bit[ slot ]= 0x03(binary 00000011)
for update
                        slot bit[ slot ]= 0x02(binary 00000010)
for delete.
          9.  pass the slot_bit array to an unload function
taking
               file,block and slot_bit array. Only unload the
rows
              that from an oracle block where for a given slot
               The slot_bit[slot] & 0x01 is true.

      10. Delete from new table TSREORG$TEMP rows is any. delete
criteria is
               file,block,slot in TSREORG$LOG
            matches file block slot
               bit[slot]&0x20 is true.
         delete from TSREORG$TEMP where rowid in
            select nrowid from TSREORG$MAP where file#=:file
            and block#=:block and
                        slot#=:slot);

          10. update TSREORG$LOG for the rows processed with
sysdate.
          11. if Window has expired release Lock. and sleep.
          12.  if Window has not expired go to next block.
          13. if no more Blocks present than
               if ( pending transactions )
                     select count(*) from V$LOCK where id1
<source object_id>
                   > 1. ( since I = reorgs current session
                   release lock and sleep.
             else
                   We are done.

          14. Apply Switch based on c.switch.
          15. Applying the switching Algorithm and completing
the reorg.
       0. Find what session id are using current reorged object.
        SQL="select s.sid , s.serial#, username,osuser,machine
from
              v$session s , v$access v
            where v.sid = s.sid and
              a.owner=<o.owner> and a.object=<c.table> and /*
not your session*/
```

【００９３】

```
                your session s.process <> getpid();"
        1.if (c.switch == IMMEDIATE) Kill any sessions connected
to source table.
                sql="alter system kill session
                goto 11.
        To obtain a list of sessions that are currently read only
for this table.
    2.if ( c.switch == NICE)
      if Any sessions are connected to source
                Release Lock and goto 9.
                sessions are connected to source. goto 11.
    11. Drop source table, rename new table to source and create
dependant objects.
    12. End Reorg cleanup.
```

【００９４】

本発明は、データベースの管理システム（ＤＢＭＳ）、特にＤＢＭＳのテーブルの再編成について説明される。しかし、ここで説明される技術は、ＤＢＭＳ、例えばスプレッドシートファイル、ワーク処理ファイル、及び他のデータ記憶装置に直接リンクされない、多くのテーブルまたはデータ記憶収納場所へ適用される。

【００９５】

本発明は、コンピュータ分野の当業者に明らかなように、本開示の技術によってプログラムされた従来の、汎用または特定のディジタルコンピュータまたはマイクロプロセッサを用いて便利に具現化することができる。

【００９６】

適切なソフトウェアコーディングは、ソフトウェア分野の当業者に明らかなように、本開示の技術に基づいて熟練したプログラマによって容易に準備することができる。本発明は、この分野の当業者に容易に明らかなように、アプリケーションの特定の集積回路の準備、または従来の要素回路の適切なネットワークを相互接続することによって、実現可能である。

【００９７】

本発明は、本発明のあらゆる処理を達成するために、コンピュータをプログラ

ムするために使用することができる記憶媒体にストアされた命令を有する記憶媒体であるコンピュータプログラム製品を含む。この記憶媒体は、フロッピー（登録商標）ディスク、光学ディスク、ＤＶＤ、ＣＤ－ＲＯＭｓ、及び磁気光学ディスクを含むあらゆる形式のディスク、ＲＯＭｓ、ＲＡＭｓ、ＥＰＲＯＭｓ、磁気または光学カード、または電子的命令を記憶するのに適したあらゆる形式のメディアを含むが、これに限定されない。

【００９８】

コンピュータ読取り可能な媒体の何れか１つに記憶される場合、本発明は、汎用の/専用のコンピュータまたはマイクロプロセッサのハードウェアを制御するための、及びコンピュータまたはマイクロプロセッサが本発明の結果を利用するユーザまたはたのメカニズムと相互作用をすることができるためのソフトウェアを含む。このようなソフトウェアは、デバイスの駆動装置、オペレーティングシステム、およびユーザアプリケーションを含むが、それらに限定されない。最後に、このようなコンピュータ読取り可能な媒体は、さらに、上述のように、少なくとも１つの選択されたデータベースのテーブル上の再編成処理を行うためのソフトウェアを含む。

【００９９】

データベーステーブル構造の識別及び検索、データベーステーブルのコピー、データブロックのアンローディングとローディング、行情報の引出、トランズアクションのロギング、ロギングされたトランズアクションの適用、及びディスプレイ、記憶、または本発明のプロセスによる結果の通信を含むが、これらの限定されない本発明の教示を実現するためのソフトウェアモジュールは、汎用の/専用のコンピュータまたはマイクロプロセッサのプログラミング（ソフトウェア）に含まれる。

【０１００】

明らかに、本発明の多くの変更及び変形が上記の教示によって可能である。したがって、請求の範囲内で、本発明は、ここに特に記載されたもの以外にも実現されることを理解すべきである。

【図面の簡単な説明】

本発明、並びにそれに伴う数多くの利点は、添付図面を参照しながら以下の詳細な説明をお読みいただけば、よく理解頂けるであろう。

【図１Ａ】

本発明によるオンライン再編成のハイレベルプロセスを示すフローチャートである。

【図１Ｂ】

同じくオンライン再編成のハイレベルプロセスを示す、図１Ａに続くフローチャートである。

【図２】

データベース・テーブルの再編成に先だって実行される初期化を示すフローチャートである。

【図３】

プロセスオブジェクトの作成を示すフローチャートである。

【図４】

テーブルのロッキングと、チェックポイントの適用とを示すフローチャートである。

【図５】

ソーステーブルのコピーの作成を示すフローチャートである。

【図６】

データブロックの構造を示す。

【図７】

ソーステーブルからソーステーブルのコピーへのデータブロックのアンロード／ロードを示すフローチャートである。

【図８】

別々のアンロード及びロード・ブロックプロセスの機能と相互作用を示すフローチャートである。

【図９Ａ】

アプライ・トランザクションプロセスを示すフローチャートである。

【図９Ｂ】

同じくアプライ・トランザクションプロセスを示す、図９Ａに続くフローチャートである。

【図１０Ａ】

データベース・テーブルのブロック毎のトランザクションの処理を示すフローチャートである。

【図１０Ｂ】

同じくブロック毎のトランザクションの処理を示す、図１０Ａに続くフローチャートである。

【図１１】

ソーステーブルを再編成されたテーブルに切り替えるプロセスを示すフローチャートである。

【図１Ａ】

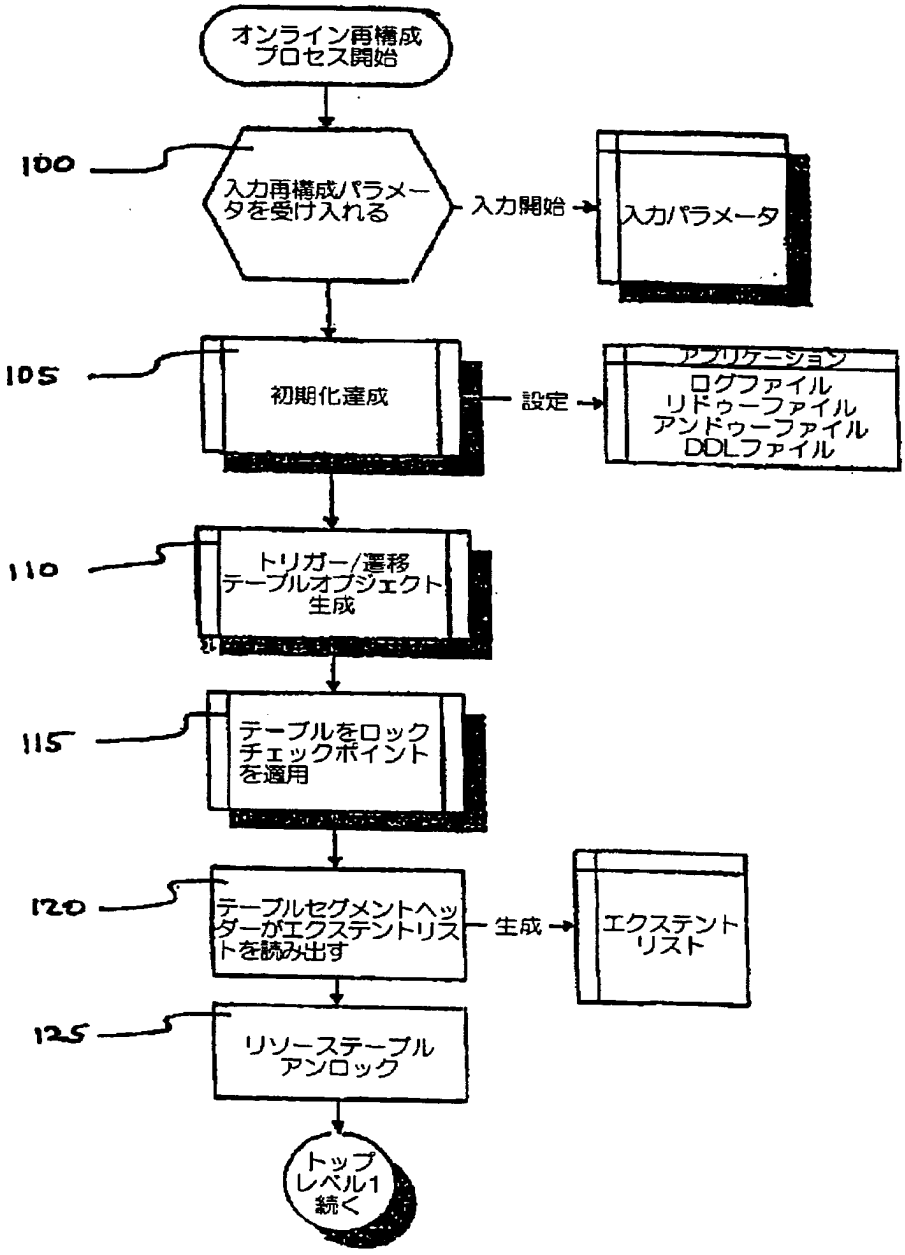

オンライン再構成プロセス開始
100
入力再構成パラメータを受け入れる
入力開始
入力パラメータ
105
初期化達成
設定
アプリケーション
ログファイル
リドゥーファイル
アンドゥーファイル
DDLファイル
110
トリガー/遷移テーブルオブジェクト生成
115
テーブルをロックチェックポイントを適用
120
テーブルセグメントヘッダーがエクステントリストを読み出す
生成
エクステントリスト
125
リソーステーブルアンロック
トップレベル1続く

FIG. 1A

【図１Ｂ】

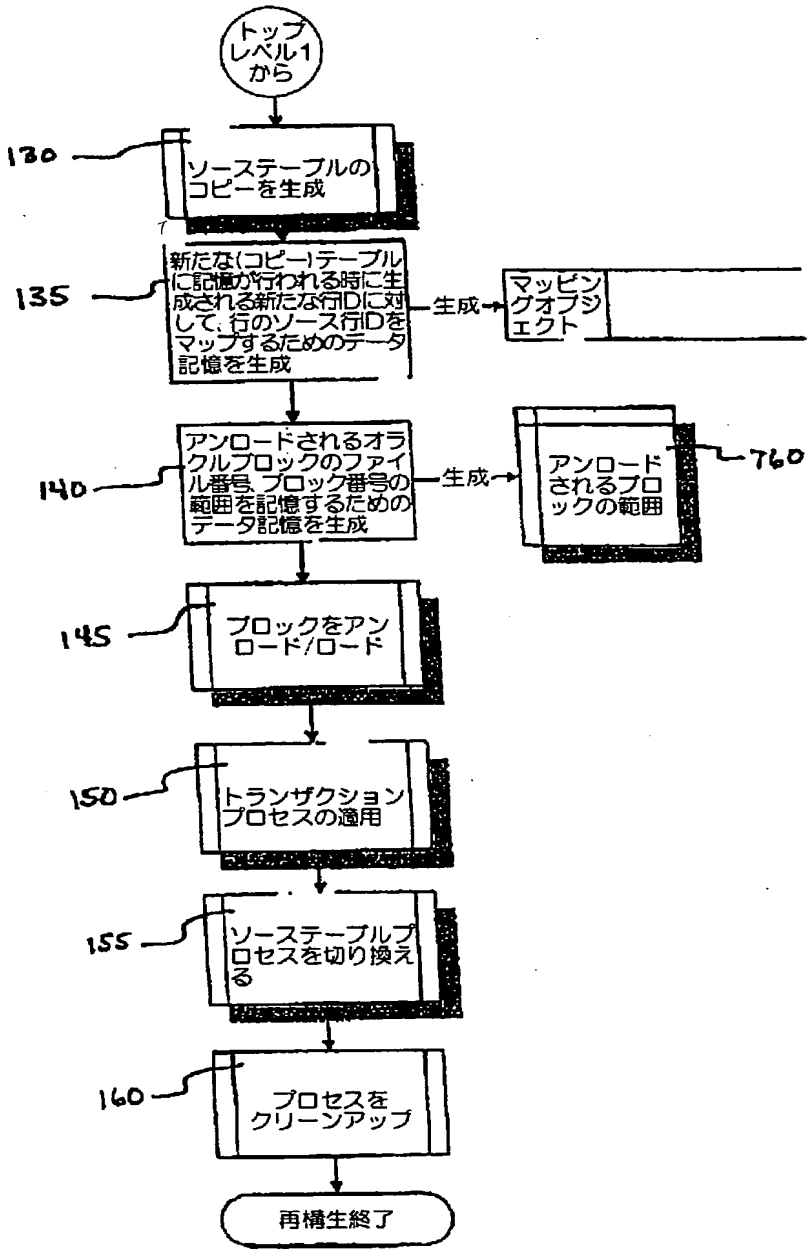

トップレベル1から
130
ソーステーブルのコピーを生成
135
新たな(コピー)テーブルに記憶が行われる時に生成される新たな行IDに対して、行のソース行IDをマップするためのデータ記憶を生成
生成
マッピングオブジェクト
140
アンロードされるオラクルブロックのファイル番号、ブロック番号の範囲を記憶するためのデータ記憶を生成
生成
アンロードされるブロックの範囲
760
145
ブロックをアンロード/ロード
150
トランザクションプロセスの適用
155
ソーステーブルプロセスを切り換える
160
プロセスをクリーンアップ
再構生終了

FIG. 1B

【図２】

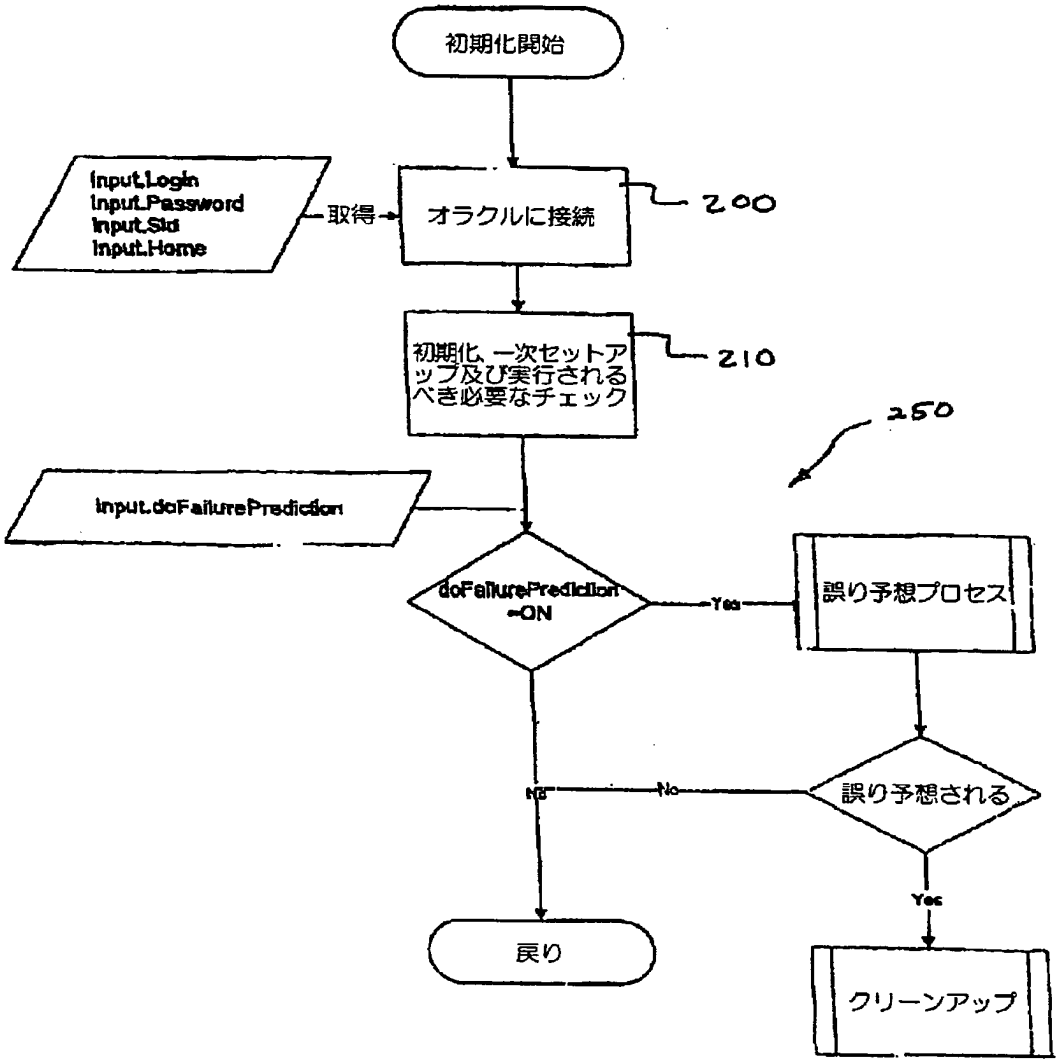

初期化開始
Input.Login
Input.Password
Input.Sid
Input.Home
取得
オラクルに接続
200
初期化、一次セットアップ及び実行されるべき必要なチェック
210
250
Input.doFailurePrediction
doFailurePrediction =ON
Yes
誤り予想プロセス
No
誤り予想される
Yes
戻り
クリーンアップ

FIG. 2

【図３】

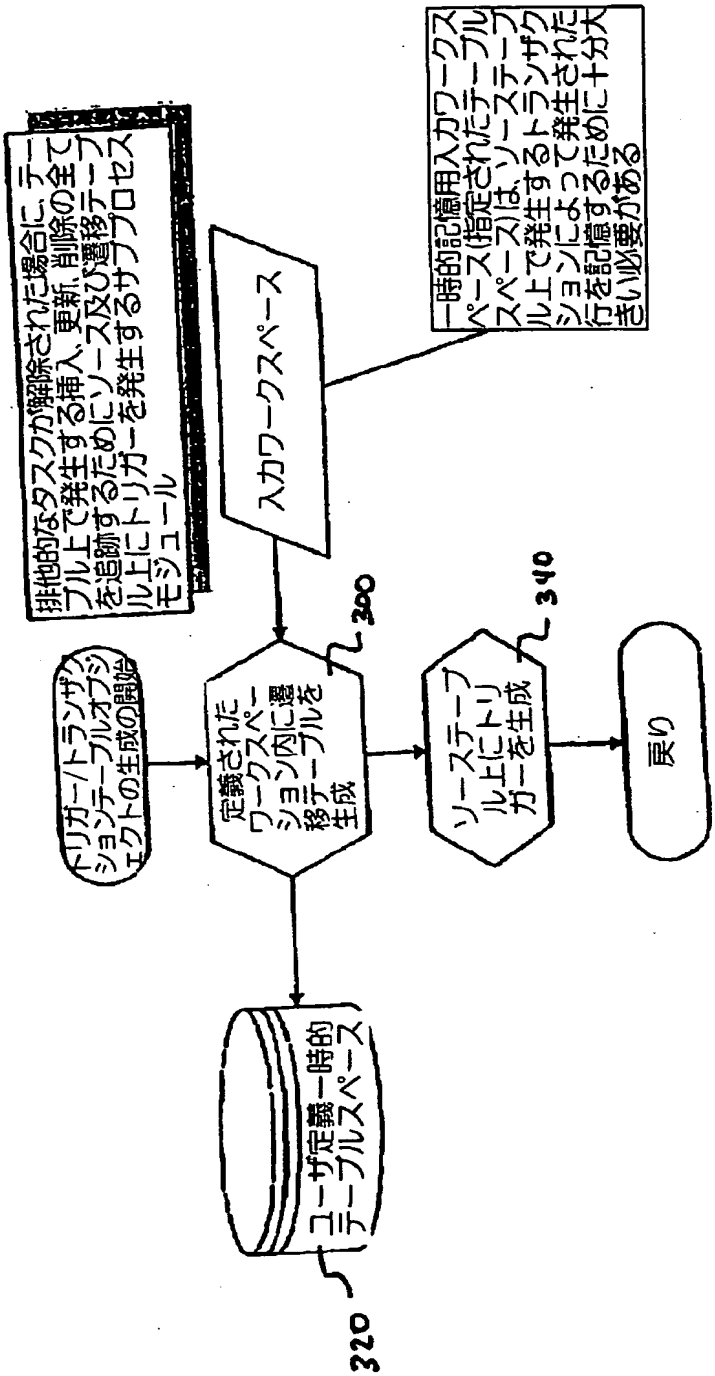

排他的なタスクが削除された場合に、テーブル上で発生する挿入、更新、削除の全てを追跡するためにソース及び遷移テーブル上にトリガーを発生するサブプロセスモジュール
入力ワークスペース
一時的記憶用入力ワークスペース(指定されたテーブルスペース)は、ソーステーブル上で発生するトランザクションによって発生するトランザクションを記憶するために十分大きい必要がある
トリガー/トランザクションテーブルオブジェクトの生成の開始
定義されたワークスペーション内に遷移テーブルを生成
300
ソーステーブル上にトリガーを生成
340
戻り
ユーザ定義一時的テーブルスペース
320

FIG. 3

【図４】

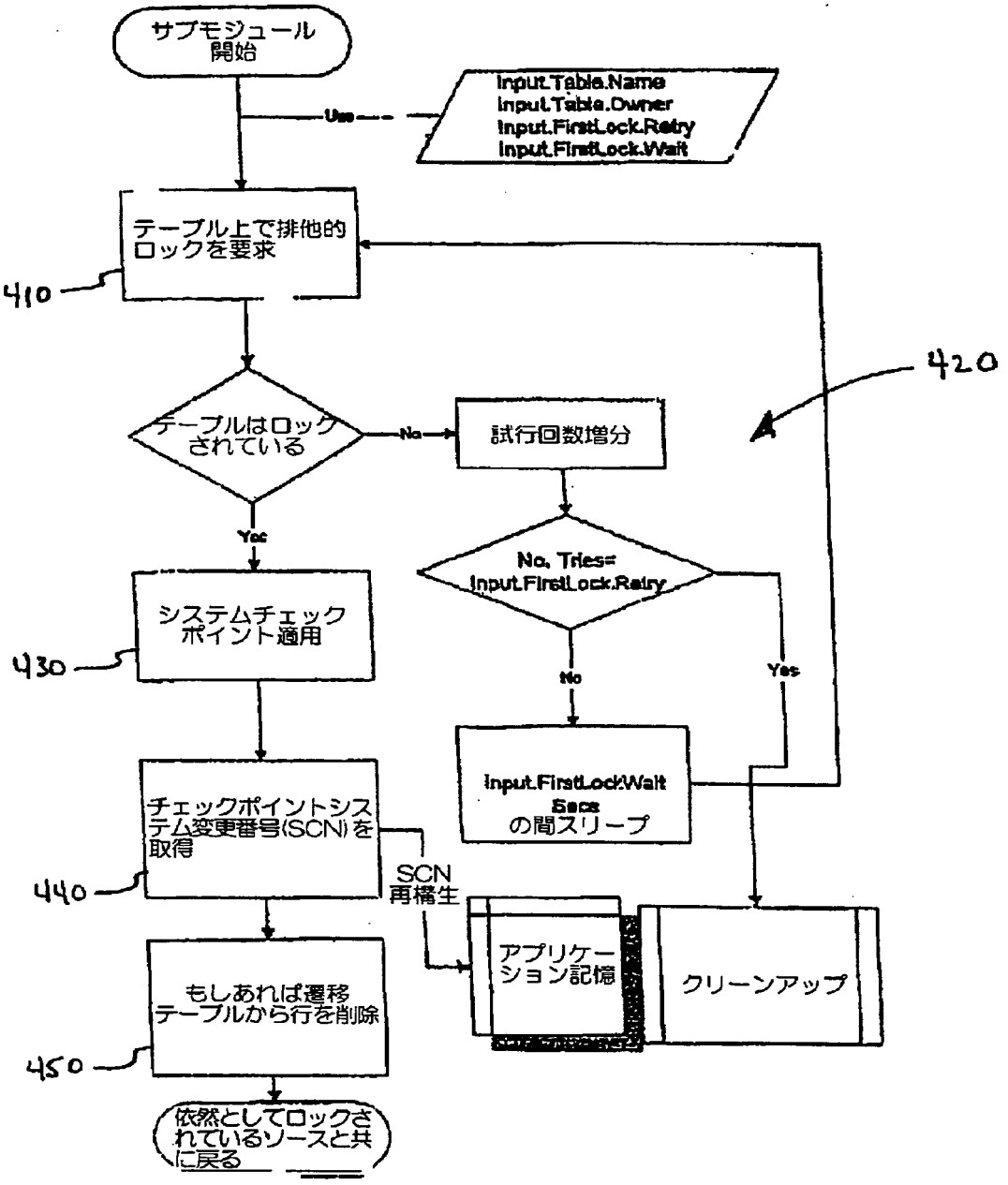

サブモジュール
開始
Use
Input.Table.Name
Input.Table.Owner
Input.FirstLock.Retry
Input.FirstLock.Wait
テーブル上で排他的
ロックを要求
410
420
テーブルはロック
されている
No
試行回数増分
Yes
No, Tries=
Input.FirstLock.Retry
No
Yes
システムチェック
ポイント適用
430
Input.FirstLock.Wait
Secs
の間スリープ
チェックポイントシス
テム変更番号(SCN)を
取得
440
SCN
再構生
アプリケー
ション記憶
クリーンアップ
もしあれば遷移
テーブルから行を削除
450
依然としてロックさ
れているソースと共
に戻る

FIG. 4

【図５】

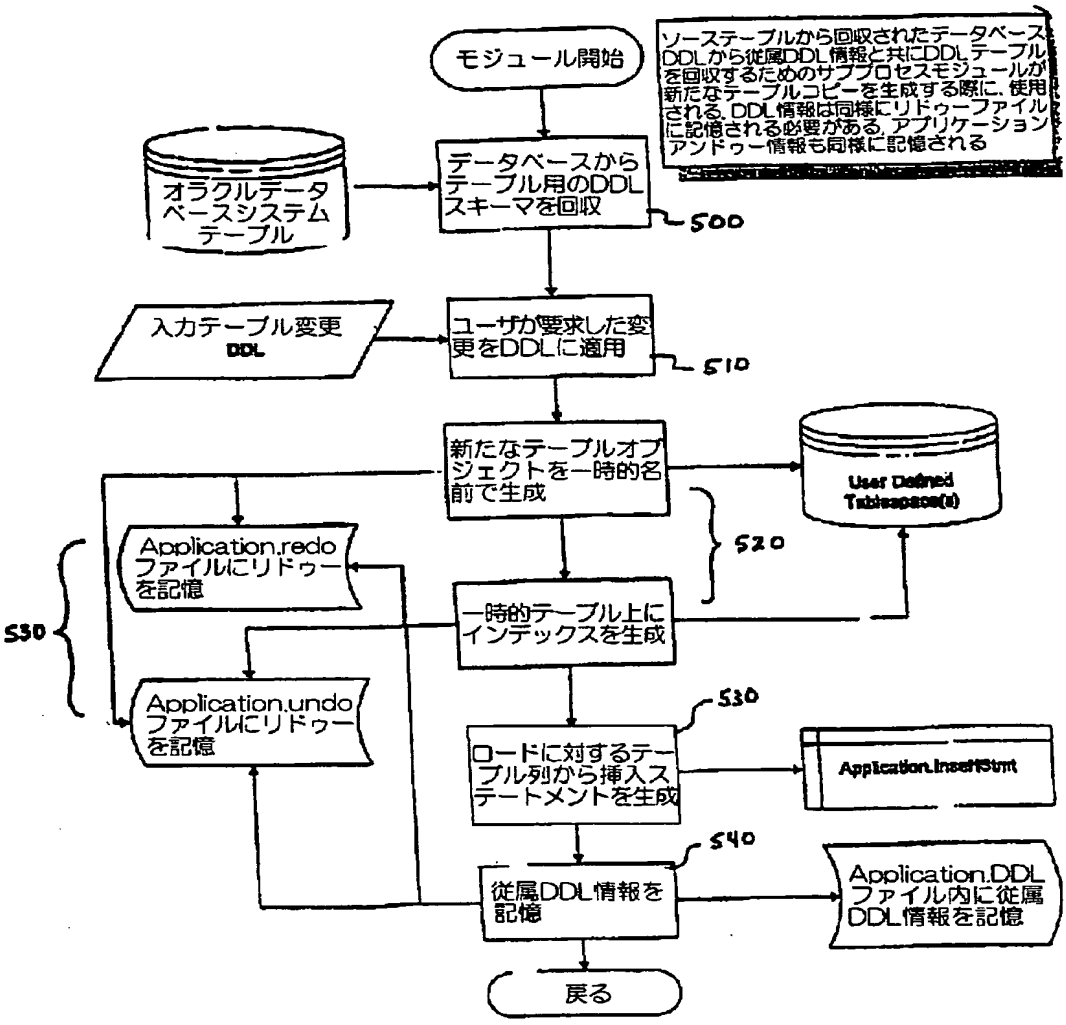

モジュール開始
ソーステーブルから回収されたデータベースDDLから従属DDL情報と共にDDLテーブルを回収するためのサブプロセスモジュールが新たなテーブルコピーを生成する際に、使用される、DDL情報は同様にリドゥーファイルに記憶される必要がある、アプリケーションアンドゥー情報も同様に記憶される
オラクルデータベースシステムテーブル
データベースからテーブル用のDDLスキーマを回収
500
入力テーブル変更 DDL
ユーザが要求した変更をDDLに適用
510
新たなテーブルオブジェクトを一時的名前で生成
User Defined Tablespace(s)
520
Application.redoファイルにリドゥーを記憶
530
一時的テーブル上にインデックスを生成
Application.undoファイルにリドゥーを記憶
ロードに対するテーブル列から挿入ステートメントを生成
Application.InsertStmt
540
従属DDL情報を記憶
Application.DDLファイル内に従属DDL情報を記憶
戻る

FIG. 5

【図６】

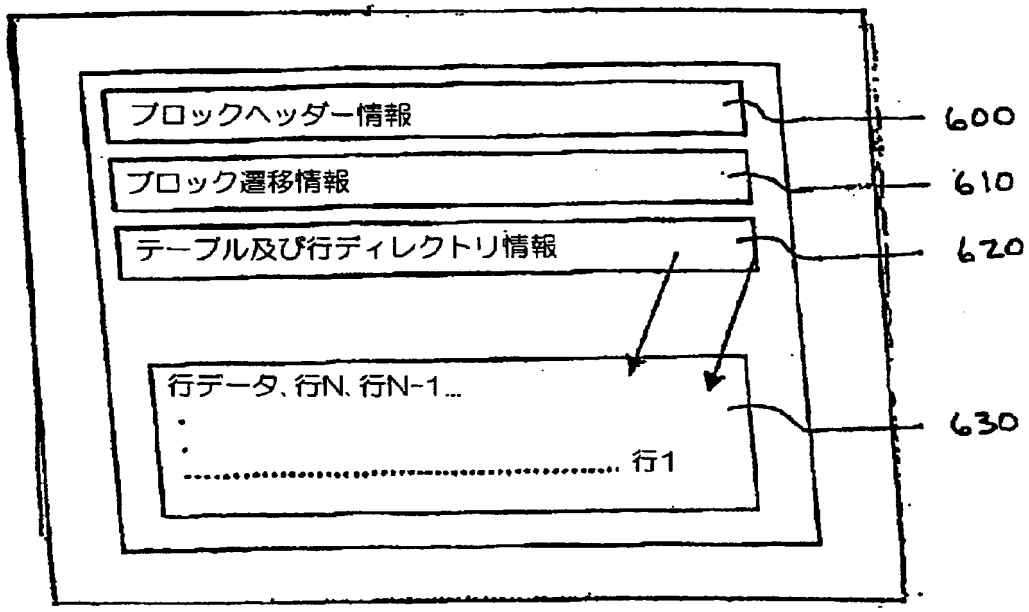

ブロックヘッダー情報
ブロック遷移情報
テーブル及び行ディレクトリ情報
行データ、行N、行N-1...
行1
600
610
620
630

FIG. 6

【図７】

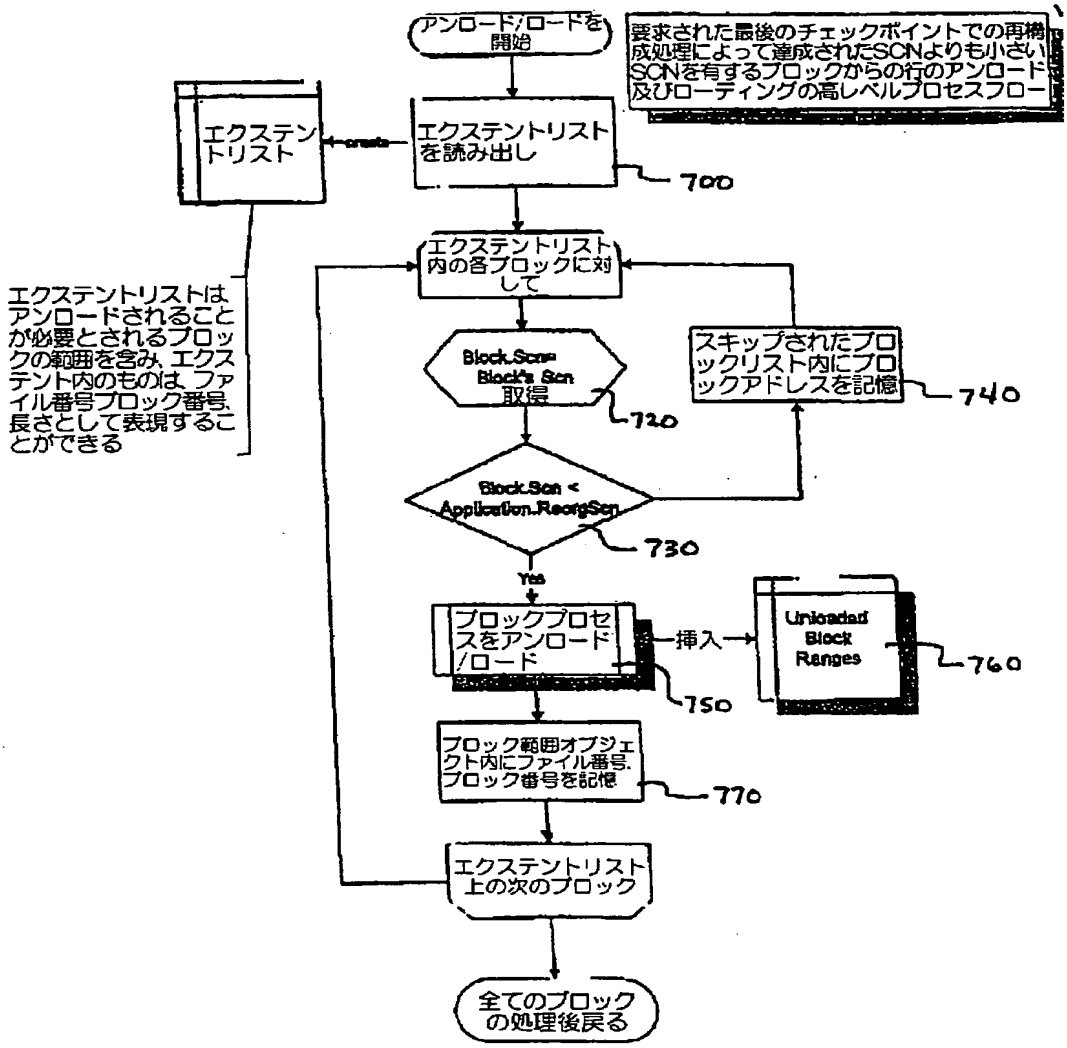

アンロード/ロードを開始
要求された最後のチェックポイントでの再構成処理によって達成されたSCNよりも小さいSCNを有するブロックからの行のアンロード及びローディングの高レベルプロセスフロー
エクステントリスト
create
エクステントリストを読み出し
700
エクステントリストはアンロードされることが必要とされるブロックの範囲を含み、エクステント内のものは、ファイル番号ブロック番号、長さとして表現することができる
エクステントリスト内の各ブロックに対して
Block.Scn= Block's Scn 取得
720
スキップされたブロックリスト内にブロックアドレスを記憶
740
Block.Scn < Application.ReorgScn
730
Yes
ブロックプロセスをアンロード/ロード
挿入
Unloaded Block Ranges
760
750
ブロック範囲オブジェクト内にファイル番号、ブロック番号を記憶
770
エクステントリスト上の次のブロック
全てのブロックの処理後戻る

FIG. 7

【図８】

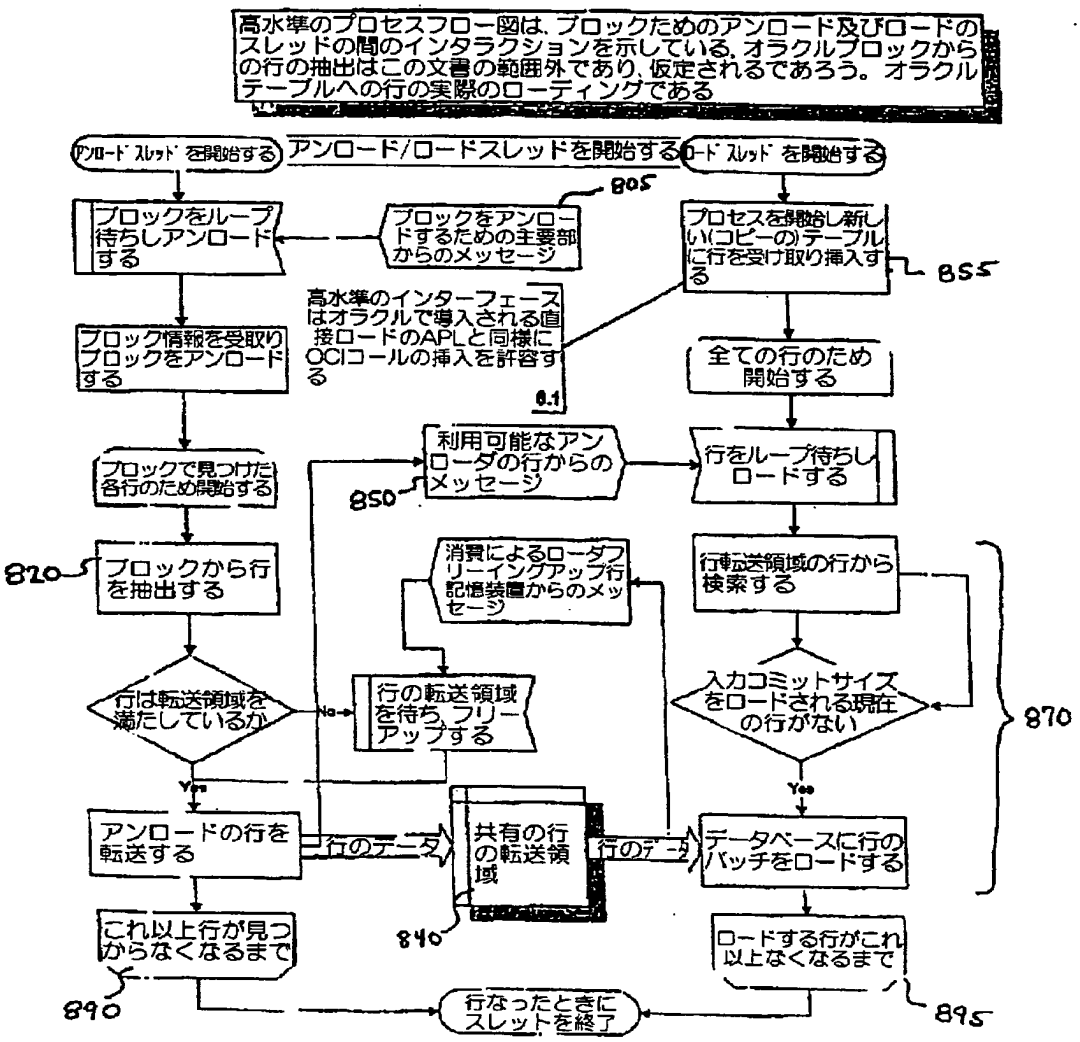

高水準のプロセスフロー図は、ブロックためのアンロード及びロードのスレッドの間のインタラクションを示している。オラクルブロックからの行の抽出はこの文書の範囲外であり、仮定されるであろう。オラクルテーブルへの行の実際のローディングである
アンロードスレッドを開始する
アンロード/ロードスレッドを開始する
ロードスレッドを開始する
805
ブロックをループ待ちしアンロードする
ブロックをアンロードするための主要部からのメッセージ
プロセスを開始し新しい(コピーの)テーブルに行を受け取り挿入する
855
高水準のインターフェースはオラクルで導入される直接ロードのAPIと同様にOCIコールの挿入を許容する
0.1
ブロック情報を受取りブロックをアンロードする
全ての行のため開始する
利用可能なアンローダの行からのメッセージ
850
行をループ待ちしロードする
ブロックで見つけた各行のため開始する
820
ブロックから行を抽出する
消費によるローダフリーイングアップ行記憶装置からのメッセージ
行転送領域の行から検索する
行は転送領域を満たしているか
No
行の転送領域を待ち、フリーアップする
入力コミットサイズをロードされる現在の行がない
870
Yes
Yes
アンロードの行を転送する
行のデータ
共有の行の転送領域
行のデータ
データベースに行のバッチをロードする
840
これ以上行が見つからなくなるまで
ロードする行がこれ以上なくなるまで
890
895
行なったときにスレットを終了

FIG. 8

【図９Ａ】

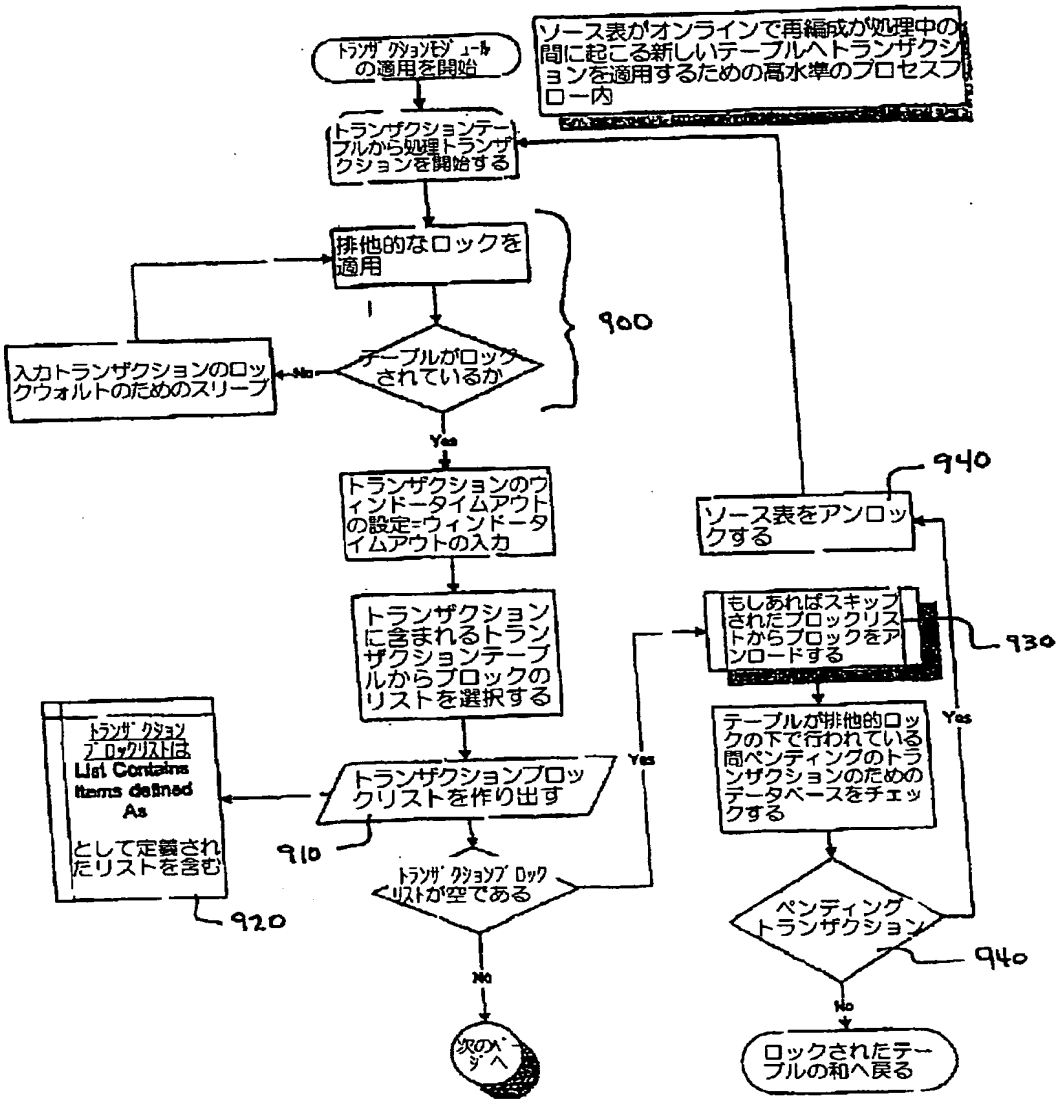

トランザクションモジュールの適用を開始
ソース表がオンラインで再編成が処理中の間に起こる新しいテーブルへトランザクションを適用するための高水準のプロセスフロー内
トランザクションテーブルから処理トランザクションを開始する
排他的なロックを適用
900
テーブルがロックされているか
No
入力トランザクションのロックウォルトのためのスリープ
Yes
トランザクションのウィンドータイムアウトの設定=ウィンドータイムアウトの入力
トランザクションに含まれるトランザクションテーブルからブロックのリストを選択する
トランザクションブロックリストは List Contains Items defined As として定義されたリストを含む
920
トランザクションブロックリストを作り出す
910
トランザクションブロックリストが空である
Yes
No
次のページへ
940
ソース表をアンロックする
もしあればスキップされたブロックリストからブロックをアンロードする
930
テーブルが排他的ロックの下で行われている間ペンディングのトランザクションのためのデータベースをチェックする
Yes
ペンディングトランザクション
940
No
ロックされたテーブルの和へ戻る

FIG. 9A

【図９Ｂ】

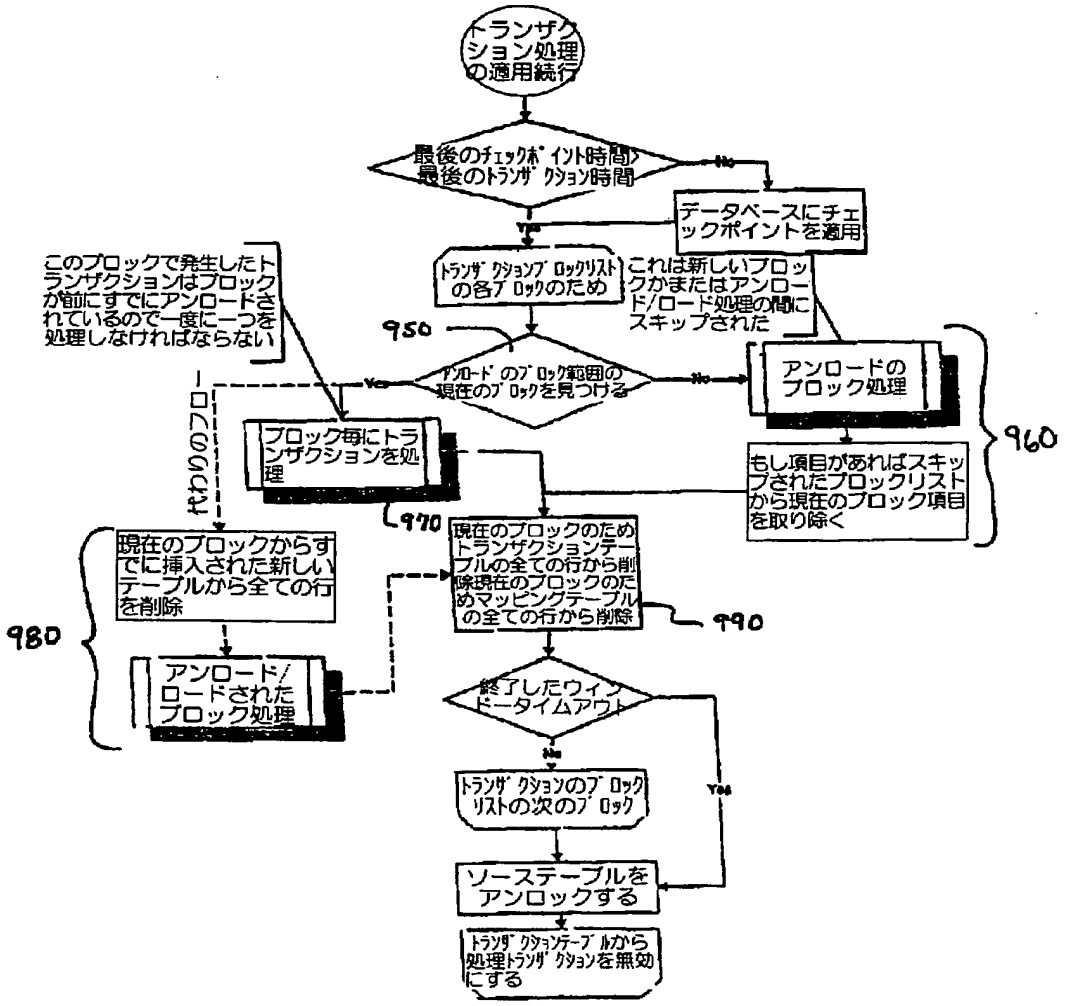

トランザクション処理の適用続行
最後のチェックポイント時間＞最後のトランザクション時間
No
Yes
データベースにチェックポイントを適用
トランザクションブロックリストの各ブロックのため
これは新しいブロックかまたはアンロード/ロード処理の間にスキップされた
このブロックで発生したトランザクションはブロックが前にすでにアンロードされているので一度に一つを処理しなければならない
950
アンロードのブロック範囲の現在のブロックを見つける
Yes
No
アンロードのブロック処理
960
ブロック毎にトランザクションを処理
970
もし項目があればスキップされたブロックリストから現在のブロック項目を取り除く
やわのフロー
現在のブロックからすでに挿入された新しいテーブルから全ての行を削除
980
現在のブロックのためトランザクションテーブルの全ての行から削除
現在のブロックのためマッピングテーブルの全ての行から削除
990
アンロード/ロードされたブロック処理
終了したウィンドータイムアウト
No
Yes
トランザクションのブロックリストの次のブロック
ソーステーブルをアンロックする
トランザクションテーブルから処理トランザクションを無効にする

FIG. 9B

【図１０Ａ】

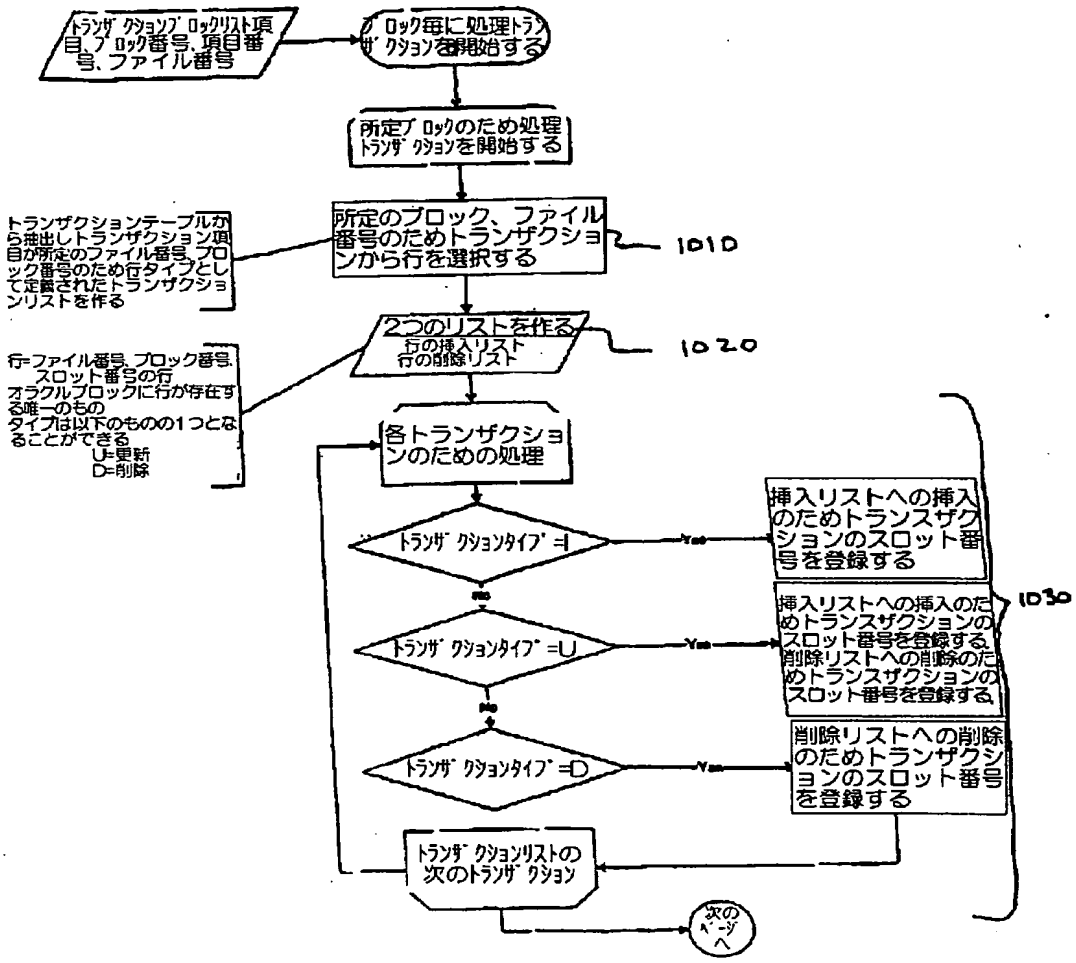

トランザクションブロックリスト項目、ブロック番号、項目番号、ファイル番号
ブロック毎に処理トランザクションを開始する
所定ブロックのため処理トランザクションを開始する
トランザクションテーブルから抽出しトランザクション項目が所定のファイル番号、ブロック番号のためタイプとして定義されたトランザクションリストを作る
所定のブロック、ファイル番号のためトランザクションから行を選択する
1010
2つのリストを作る
行の挿入リスト
行の削除リスト
1020
行=ファイル番号、ブロック番号、スロット番号の行
オラクルブロックに行が存在する唯一のもの
タイプは以下のものの1つとなることができる
U=更新
D=削除
各トランザクションのための処理
トランザクションタイプ=I
Yes
挿入リストへの挿入のためトランスザクションのスロット番号を登録する
No
トランザクションタイプ=U
Yes
挿入リストへの挿入のためトランスザクションのスロット番号を登録する
削除リストへの削除のためトランスザクションのスロット番号を登録する
1030
No
トランザクションタイプ=D
Yes
削除リストへの削除のためトランザクションのスロット番号を登録する
トランザクションリストの次のトランザクション
次のページへ

FIG. 10A

【図１０Ｂ】

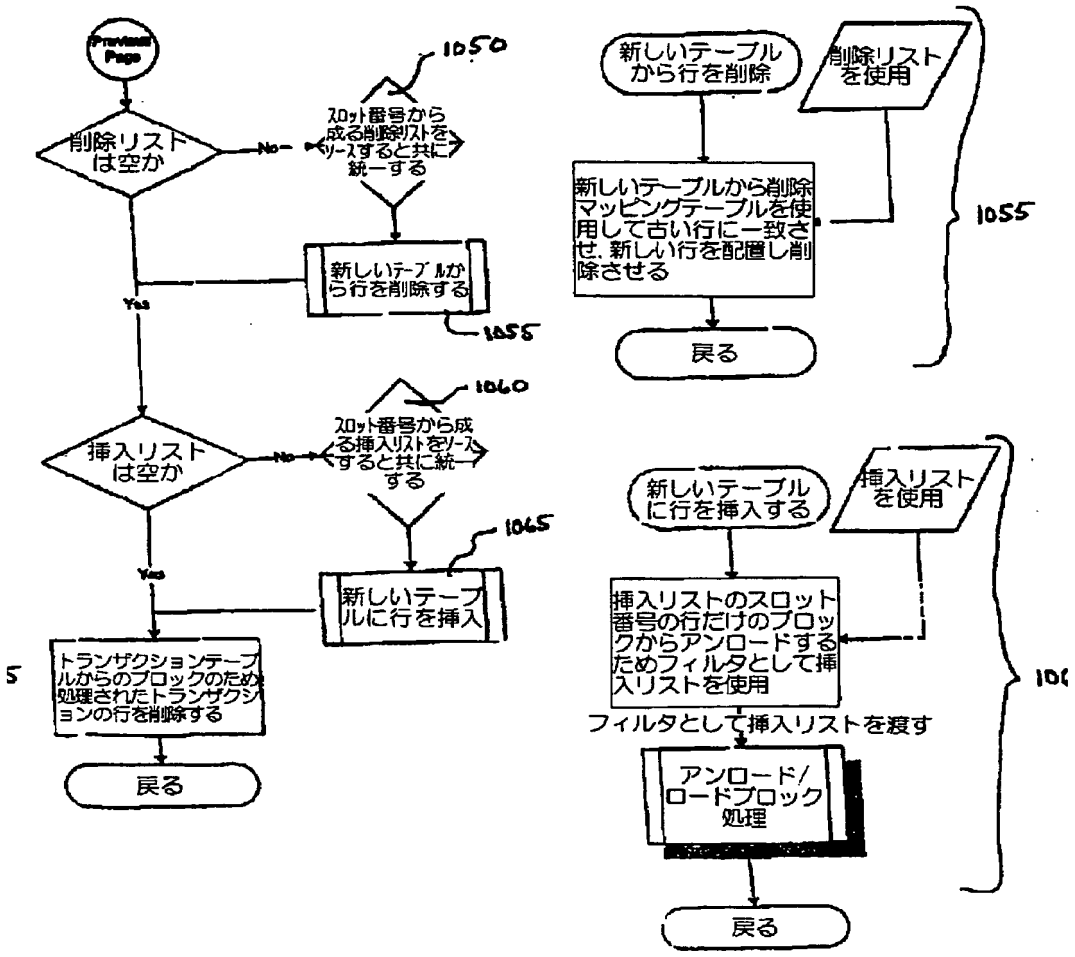

削除リストは空か
No
スロット番号から成る削除リストをソースすると共に統一する
1050
新しいテーブルから行を削除する
1055
Yes
挿入リストは空か
No
スロット番号から成る挿入リストをソースすると共に統一する
1060
新しいテーブルに行を挿入
1065
Yes
トランザクションテーブルからのブロックのため処理されたトランザクションの行を削除する
戻る
新しいテーブルから行を削除
削除リストを使用
新しいテーブルから削除マッピングテーブルを使用して古い行に一致させ、新しい行を配置し削除させる
1055
戻る
新しいテーブルに行を挿入する
挿入リストを使用
挿入リストのスロット番号の行だけのブロックからアンロードするためフィルタとして挿入リストを使用
フィルタとして挿入リストを渡す
アンロード/ロードブロック処理
戻る

FIG. 10B

【図１１】

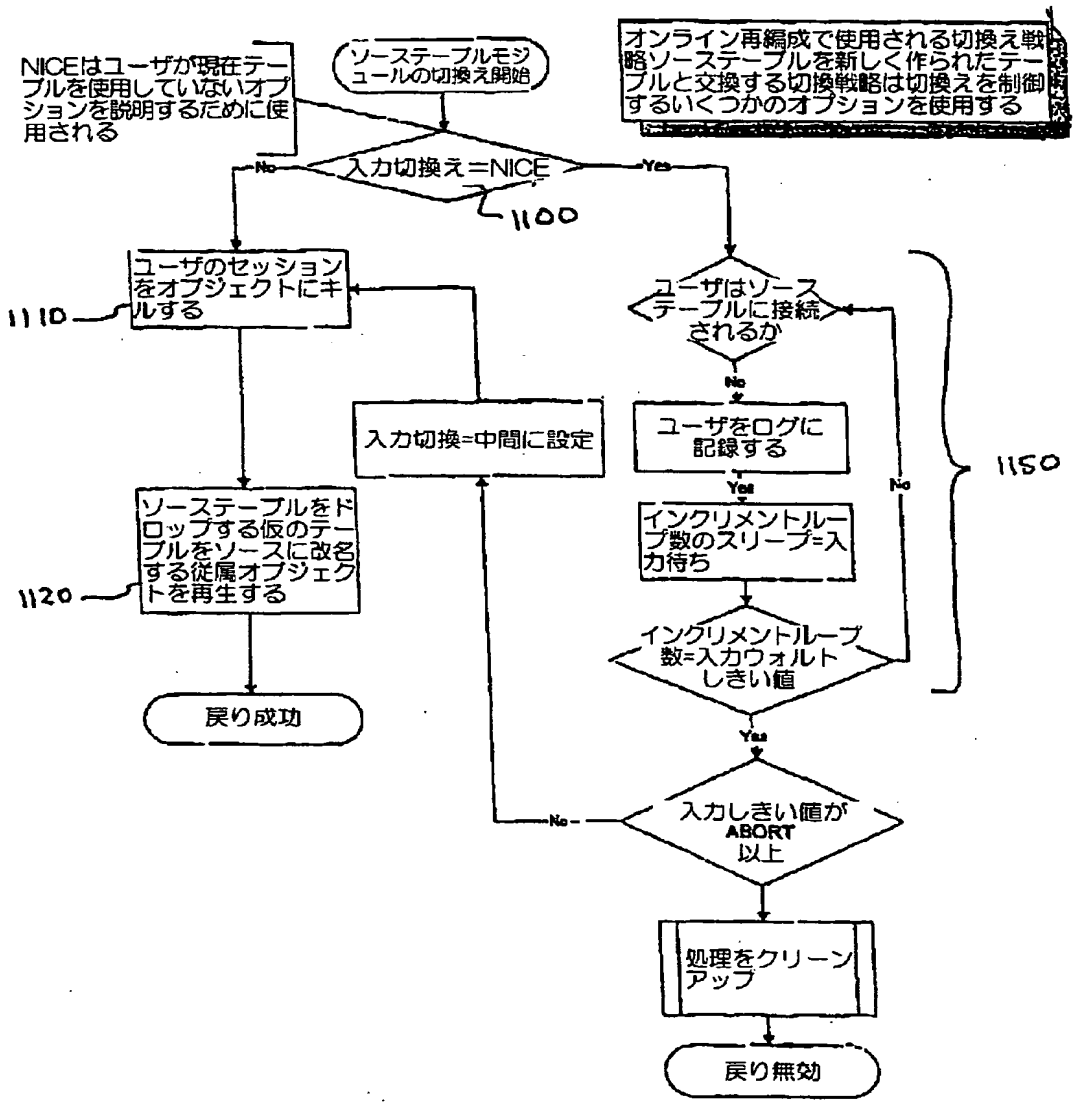

NICEはユーザが現在テーブルを使用していないオプションを説明するために使用される
ソーステーブルモジュールの切換え開始
オンライン再編成で使用される切換え戦略ソーステーブルを新しく作られたテーブルと交換する切換戦略は切換えを制御するいくつかのオプションを使用する
入力切換え=NICE
No
Yes
1100
ユーザのセッションをオブジェクトにキルする
1110
ユーザはソーステーブルに接続されるか
No
ユーザをログに記録する
Yes
インクリメントループ数のスリープ=入力待ち
入力切換=中間に設定
No
1150
ソーステーブルをドロップする仮のテーブルをソースに改名する従属オブジェクトを再生する
1120
インクリメントループ数=入力ウォルトしきい値
Yes
戻り成功
No
入力しきい値がABORT以上
処理をクリーンアップ
戻り無効

FIG. 11

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/22044

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G06F 17/30
US CL :707/100, 200, 201
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 707/100, 200, 201

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WEST, IEEE

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,553,279 A (GOLDRING ET AL.) 03 September 1996, the entire paper is relevant.	1, 11, 21, and 31
Y	US 5,721,915 A (SOCKUT ET AL.) 24 February 1998, the entire paper is relevant.	1, 11, 21, and 31
Y	US 5,781,903 A (RUSTERHOLZ) 14 July 1998, the entire paper is relevant.	I, 11, 21, and 31
Y	Oracle 7tm Server, "Concepts Manual", December 1991, pages 15-1 to 15-14	2, 12, 20, 22, 30, and 32

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 27 NOVEMBER 1999	Date of mailing of the international search report 23 DEC 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer THUY PARDO James R. Matthews Telephone No. (703) 305-1091

Form PCT/ISA/210 (second sheet)(July 1992)*